PRAGMA

A publication of Semaphore Corporation

Issue Number 3 February, 1983

IN THIS ISSUE

DEPARTMENTS

Is Pragma a Rare Medium, Well Done?

It seems that no publication can go very long before formally polling its readers for comments and criticisms. Pragma is no exception. On the right is a questionnaire for you, our readers. We invite you all to answer as many of the questions as you can, and to return the form to us. We'll organize and tabulate the results and present them in the next issue. Although Pragma has rather rigorously adhered to the same general format and content for all three initial issues, there is always room for improvement. We hope the survey will indicate which areas you would like to see changed, or expanded, or even dropped. We will incorporate your suggestions into the publication as quickly as possible.

• • •

Many pieces of mail cross our desks each month (especially now that Prime has us on their mailing list — we seem to get a press release from Prime almost once a day!). Perhaps the most interesting piece of mail received in the past few weeks has been the first edition of the IDBMA directory. The IDBMA is the International Database Management Association, which is the group organizing the March 1983 Tahoe conference for Pick users. Their impressive little directory has been nicely done. It contains an alphabetical list of IDBMA members and their addresses, a list of local user group presidents, lists of software packages by application and by name, and a list of software vendors.

• • •

We're interested in hearing from any users who have performed any type of evaluation as part of deciding whether or not to acquire a software product. If you have compared two similar software products to determine which one best met your needs, or if you feel you can describe the strong and weak points in a software package you have purchased or only reviewed or investigated, or even if you simply have comments about the range in the quality of documentation you have received from different software vendors, please write or call Pragma so that we can discuss your findings.

• • •

We're also interested in hearing from any reader who claims to be using the smallest or the largest hardware configuration running the Pick operating system (or a Pick look-alike). Recent developments at both the high and low end of the hardware scale have piqued our curiousity. Is the Pick operating system used on a greater range of hardware configurations than any other operating system? If you think you hold the record for the smallest or largest computer running Pick software, let us know so we can present the current record holder in the next Pragma. A configuration's size will be judged by the number of ports, amount of disk, and amount of memory.

—The Editors

Pragma Reader Survey

How do you rate the regular departments that appear in each issue of Pragma? Check the boxes below:

	Poor	Fair	Good	Excellent
Utilities	☐	☐	☐	☐
User Profile	☐	☐	☐	☐
Benchmarks	☐	☐	☐	☐
Wish List	☐	☐	☐	☐
Command Files	☐	☐	☐	☐
Queries	☐	☐	☐	☐
The Computer Room	☐	☐	☐	☐
Letters	☐	☐	☐	☐
Games	☐	☐	☐	☐

What would you like to see more of in each issue of Pragma?

_____

_____

_____

What would you like to see less of in each issue of Pragma?

_____

_____

_____

What brands of Pick-based systems do you use?

_____

What is your job title? _____

Please give us any other comments you have about Pragma:

_____

_____

_____

Please return this questionnaire to Pragma, 207 Granada Drive, Aptos, CA 95003.

PRAGMA

Issue Number 3 February, 1983

Pragma is published at least four times a year by

Semaphore Corporation
207 Granada Drive
Aptos, California 95003

SUBSCRIPTIONS

Subscriptions are $25 per issue, $100 per year (four issues) in the USA, $38 per issue to other countries by airmail. All payments must be in US dollars drawn on a US bank.

Paid subscribers who refer new subscribers will receive a free subscription extension of one issue for each referral. Such referrals must include an appropriate subscriber number as explained on subscription order forms.

Address all subscription correspondence to the **Pragma** Circulation Manager, **Semaphore Corporation.** When writing, enclose your issue's mailing address label or the numbers from the upper right corner of your label.

Address changes should be sent at least four weeks in advance. Include old and new addresses along with your issue's mailing address label.

SUBMITTALS

All correspondence and material received will be considered for publication. Except for correspondence used in the Letters Department, authors are paid up to $200 per full published page for submitted material used in **Pragma.** Actual payment amounts are decided by the Editors and vary with the amount of required editing and rework and with the length of each submittal. Authors are also granted free subscription extensions of one issue for submittals of at least one full published page. Address all submittals and correspondence to the **Pragma** Editors, **Semaphore Corporation.** All letters to the Editors are welcome and as many as possible will be published in the Letters Department.

All submittals must be typed on white paper and double spaced. The first page of any submittal and any accompanying material must include the author's name, address, telephone number and the date. All pages must be numbered.

Hand-typed program listings and other simulated printouts will not be accepted. Authors should submit actual computer-generated output. Print all listings with a fresh black ribbon on continuous white paper. Do not print on perforations. Do not include page numbers or headings on listings in order to simplify reduction and layout. Accompanying documentation should refer to listings by content, line number or symbolic labels, not by page number.

Drawings, schematics and other illustrations must be in black ink on white paper and drawn in a large scale to allow significant reduction. Photographs must be black and white glossies.

Manuscripts are submitted at the author's risk. Unused manuscripts will be returned if a stamped, self-addressed envelope is included. Requests to review galleys must accompany the manuscript when it is first submitted. The Editors reserve the right to edit all submittals.

ADVERTISING

Send all advertising correspondence, requests for advertising rates, and advertising copy to the **Pragma** Advertising Manager, **Semaphore Corporation.** Advertisers sending press releases are requested to telephone the Advertising Manager for an interview at 408-688-9200 within two weeks after submitting the material.

READER SERVICES

Is there a service **Semaphore Corporation** can provide for you? Address all inquiries to **Pragma** Reader Service, **Semaphore Corporation,** or telephone 408-688-9200.

Edit Aides

by Michael D. Rossetti
Sandy, UT

Edit aides are procs that emulate frequently used commands, but which additionally remember and recall default file and item names.

The dialogue at right demonstrates the use of a facility which remembers item and file names so that they don't have to be repetitively entered time after time in a series of commands.

The proc listings beginning on page 9 provide all of the following commands:

- **BC** Compile a program. If it compiles with no errors then catalog the program.
- **E** Edit an item. Tab stops are set and the prestored P command is set to "L22".
- **P** Print an item.
- **R** Run a program.
- **X** Execute a cataloged program.

Each of the above commands will automatically remember the most recent item and/or file name that was previously input with any of the commands.

The Master Dictionary containing these procs (which have been written for a Microdata system) must also contain the two items named AUTO-EDIT-NAMES and MAKE-AUTO-EDIT. AUTO-EDIT-NAMES is where the last used file and item names are stored. MAKE-AUTO-EDIT, which is called by each command proc to retrieve the last used names, reads and writes the AUTO-EDIT-NAMES item.

The procs shown here can be used as guides for creating other similar commands that remember file and item names, and not just for items that happen to be programs. For example, a command named RO can be created for executing Runoff while automatically providing a document's file and item names.

In this sample dialogue, the user begins with the E (for Edit) command, and explicitly names the file and item to be edited. The next commands (BC and P) automatically recall the file and item names for the user. When the user again gives an E command, but only specifies the item name, the file name is still automatically supplied.

```
:E BP PRINT
EDIT BP PRINT
TOP
.G59
059                PRINT LINE
.L3
060              END
061              I=I+5
062          REPEAT
.G61
061              I=I+5
.R/5/1
061              I=I+1
.FI
'PRINT' FILED.

:BC
BASIC AND CATALOG BP PRINT
**************************************
**************************************
****
LINE 64 [BO]  COMPILATION COMP
LETED

[241] 'PRINT' CATALOGED; 2 FRA
MES USED.

:P
PRINT BP PRINT
ENTRY # 009

:E REPROJECT
EDIT BP REPROJECT
TOP
.G15
015          DIM POSS(25)
```

Some New Subscribers

George Lopatin
International Envelope
Aston, PA

Robert W. Murray
Pryor Fabrics Inc.
Brea, CA

John D. Vetter
Delta Data Consultants Inc.
Shreveport, LA

Joel Nirenberg
KTS Business Systems Inc.
Toronto, ON

Bruce Corbett
Tab Products Co.
Palo Alto, CA

Frank E. Wiley
Grid Systems Corp.
Mt. View, CA

EDIT AIDES

```
    BC
001 PQN
002 C
003 C THIS PROC WILL GET A
004 C FILE AND ITEM NAME,
005 C COMPILE IT, AND CATALOG
006 C IT IF THE COMPILE WAS
007 C SUCCESSFUL.
008 C
009 MV %1 "BASIC AND CATALOG"
010 [MD MAKE-AUTO-EDIT]
011 RO
012 C BUILD COMPILE COMMAND.
013 NHBASIC
014 H%2
015 H
016 H%3
017 P
018 C SEE IF COMPILE WAS GOOD.
019 IF E # BO X- CATALOG NOT P
    ERFORMED -
020 C BUILD CATALOG COMMAND.
021 NHCATALOG
022 H%2
023 H
024 H%3
025 P

    E
001 PQN
002 C
003 C THIS PROC WILL GET THE
004 C FILE NAME AND ITEM NAME
005 C TO BE EDITED.
006 C
007 MV %1 "EDIT"
008 [MD MAKE-AUTO-EDIT]
009 RO
010 C SET UP THE EDIT COMMAND.
011 NHEDIT
012 NA2
013 NA3
014 STON
015 C SET THE PRESTORED COMMAN
    D TO DO A 'L22'
016 NHP L22<
017 C SET THE TAB STOPS.
018 NHTB 8,11,14,17,20,23,26,2
    9,32,35,38,41,44,47,50,53,
    56,59,61
019 STOFF
020 P

    P
001 PQN
002 C
003 C PRINT THE ITEM IN A FILE
004 C USING THE SUPPLIED OR
005 C STORED NAMES.
006 C
007 MV %1 "PRINT"
008 [MD MAKE-AUTO-EDIT]
009 RO
010 C BUILD COPY TO PRINTER
011 C COMMAND.
012 NHCOPY
013 NA2
014 NA3
015 H (P)
016 P

    R
001 PQN
002 C
003 C THIS PROC WILL RUN A
004 C BASIC PROGRAM USING THE
005 C SUPPLIED OR STORED FILE
006 C AND ITEM NAMES
007 C
008 MV %1 "RUN"
009 [MD MAKE-AUTO-EDIT]
010 RO
011 C BUILD THE RUN COMMAND
012 NHRUN
013 NA2
014 NA3
015 P

    X
001 PQN
002 C
003 C THIS PROC WILL EXECUTE A
004 C BASIC PROGRAM USING THE
005 C SUPPLIED OR STORED ITEM
006 C NAME.
007 C
008 MV %1 "EXECUTE"
009 [MD MAKE-AUTO-EDIT]
010 RO
011 C JUST EXECUTE ITEM NAME
012 C AS IF IT WERE A VERB.
013 NA3
014 C TELL IT TO ENTER DEBUG
015 C IF AN ERROR IS DETECTED.
016 H (F)
017 P
```

EDIT AIDES CONTINUED

```
    MAKE-AUTO-EDIT
001 PQN
002 C
003 C THIS PROC IS CALLED AS A
004 C SUBROUTINE BY OTHER
005 C QUICK EDIT PROCS TO
006 C PROVIDE AND UPDATE THE
007 C DEFAULT FILE AND ITEM
008 C NAMES.
009 C
010 C OPEN THE MASTER DICT.
011 C
012 F-CLEAR 1
013 F-OPEN 1 MD
014 XFAILED TO OPEN YOUR MASTE
    R DICTIONARY
015 C
016 C READ THE FILE AND ITEM
017 C NAMES USED THE LAST TIME
018 C THIS PROC WAS CALLED.
019 C
020 F-READ 1 AUTO-EDIT-NAMES
021 C
022 C SET UP THE ITEM ID FOR
023 C WHEN THE AUTO-EDIT-NAMES
024 C ITEM IS UPDATED.
025 C
026 MV &1.0 "AUTO-EDIT-NAMES"
027 C
028 C SEE IF THE OPERATOR
029 C SUPPLIED BOTH A FILE
030 C NAME AND AN ITEM NAME,
031 C IN WHICH CASE NONE OF
032 C OLD ONES WILL BE USED.
033 C
034 IF %2 IF %3 GO 10
035 C
036 C SEE IF NO NEW NAMES WERE
037 C GIVEN AT ALL, IF SO BOTH
038 C OLD ONES WILL BE USED.
039 C
040 IF # %2 GO 10
041 C
042 C IF YOU GET HERE THEN A
043 C ITEM NAME WAS GIVEN BUT
044 C THEN FILE NAME WASN'T.
045 C PUT THE ITEM NAME INTO
046 C %3 AND CLEAR %2.
047 C
048 MV #1 %2
049 RI2
050 MV %3 #1
051 C
052 C BY NOW, ANY SUPPLIED
053 C ITEM NAME IS IN %3 AND
054 C FILE NAME IN %2.  IF
055 C EITHER ONE IS GIVEN THEN
056 C MOVE IT INTO THE RECORD
057 C TO BE WRITTEN TO
058 C AUTO-EDIT-NAMES IN MD.
059 C
060 10 IF %2 MV &1.1 %2
061 IF %3 MV &1.2 %3
062 C
063 C AT THIS POINT THE RECORD
064 C BUFFER IS GUARANTEED TO
065 C HAVE THE CORRECT NAMES
066 C IN IT.  MOVE THEM INTO
067 C THE PRIMARY INPUT BUFFER
068 C
069 MV %2 &1.1
070 MV %3 &1.2
071 C
072 C DISPLAY THE COMMAND
073 C GIVEN BY THE CALLING
074 C PROC AND THE NAMES
075 C FINALLY CHOSEN SINCE
076 C WE WANT THE OPERATOR TO
077 C KNOW WHAT WAS PREVIOUSLY
078 C USED.
079 C
080 D1+
081 0 +
082 D2+
083 0 +
084 D3
085 C
086 C UPDATE AUTO-EDIT-NAMES
087 C ITEM SO THAT NEXT CALL
088 C TO THIS PROC REMEMBERS
089 C WHAT WAS USED THIS TIME.
090 C
091 F-WRITE 1
092 RTN
```

```
    AUTO-EDIT-NAMES
001 PROGRAM-FILE
002 PAYROLL.RECONCILE
```

utilities

Utility programs have saved many a programmer from the less appealing aspects of software development and maintenance. Reformatting code, stripping files clean of control characters, converting data from one format to another — all are examples of tasks best delegated to a program and not a programmer.

Good programmers will collect a "toolbox" of utility software to use from time to time. If you have a useful utility, send it in for publication. A regular feature in Pragma will be this Utilities Department, where good software tools will be spotlighted.

LIST.POINTERS: A Proc for Cross-Referencing Q-Pointers

A proc for generating a cross-reference listing showing all references to files by Q-pointers is presented. The proc automatically scans the Master Dictionary of each account and builds a file that is then sorted to show all files referenced by Q-pointers, and the names and locations of all Q-pointers found.

A Q-pointer (more formally called a "file synonym definition item") is a convenient mechanism that allows a Master Dictionary to name a file that is actually defined and allocated with a D-pointer (or "file definition item") by some other Master Dictionary. Q-pointers are also often used to create various different names for one file defined in the same Master Dictionary as the Q-pointers.

Once a Q-pointer is located, it is easy to immediately determine for which file the Q-pointer is a synonym, since the account and file names being referenced are stored in attributes two and three of the Q-pointer. But finding synonyms for a given file name is more complicated: an exhaustive search of all Master Dictionaries must be made to find all Q-pointers that reference the file.

LIST.POINTERS is a proc that automatically cross-references all uses of Q-pointers in a system. For systems with many accounts, LIST.POINTERS is a handy tool for quickly showing which accounts use files in other accounts.

Figure 1 is a typical listing generated by the LIST.POINTERS proc. In this example, we see that the BP file in the ENG account is referenced by a Q-pointer named ENG.BP in the DEVELOP account and by one named ENGBP in the SP account, while the VM file in the PROGS account can be accessed by the name VENDOR.MASTER from the USERS account.

Figure 2 is the source for the LIST.POINTERS proc, which was written for a Microdata system. Lines 2 through 4 of the proc guarantee that LIST.POINTERS is being executed from the SP or SYSPROG account by using the U50BB mode call as described on page 26 of Pragma #2. LIST.POINTERS can be changed to execute from any account by editing lines 4, 15 and 20, but since LIST.POINTERS accesses the SYSTEM file and requires its own Master Dictionary to contain Q-pointers to every account Master Dictionary, it is usually most convenient to execute LIST.POINTERS from SP or SYSPROG as shown here.

LIST.POINTERS uses a scratch file called QREFS for saving the descriptions of the Q-pointers it finds. Each QREFS item has an item identifier consisting of an account name on the left concatenated with a file name on the right, with an asterisk used as a separator between the two names. The identifier indicates a file being referenced by Q-pointers. Attribute one of each QREFS item is a multivalued list of all Q-pointers that reference the identifier. Each entry in the list is in the same account-asterisk-file format. Since QREFS is re-used as a scratch file each time LIST.POINTERS is run, simultan-

eously executing LIST.POINTERS under the same account from two different ports will cause the QREFS file to contain incorrect results.

Line 5 opens the QREFS file to buffer 1 for later reading and writing by lines 31 and 36. Line 6 is the error return abort for the F-OPEN in line 5. Lines 7 and 8 clear QREFS of the results from any previous execution of LIST.POINTERS. Lines 9 through 12 select all account names and save them in select register 1 for retrieval one by one at line 13. This is where LIST.POINTERS requires access to the SYSTEM file. Note that the SSELECT tests for the presence of data (assumed to be the account creation date) in attribute 14 of each account definition item, so that only "real" accounts are found and items in the SYSTEM file like BLOCK-CONVERT and POINTER-FILE are skipped. Accounts that were created with the standard CREATE-ACCOUNT proc should have the creation date stored in attribute 14. If an installation's SYSTEM file does not follow this convention, all account items can be edited to conform, or line 9 in LIST.POINTERS should be changed to use whatever criteria will cause the successful selection of all account definition items.

The search for Q-pointers in each account's Master Dictionary begins with the retrieval of the next account name from select register 1 in line 13. If the list of account names is exhausted,

Some New Subscribers

Robert Schmidt
C.A. Lawton Co.
De Pere, WI

Monisa Trice
Drs. Groover Christie & Merritt
Silver Spring, MD

Peter McComber
Pacific Brewers Distributors Ltd.
Burnaby, BC

Kirby Plumlee
City of Garden Grove
Garden Grove, CA

```
ACCOUNT*FILE. . . . . .  POINTED TO BY ACCOUNT*FILE. . .

ACC*MD                   DEVELOP*ACC
                         DEVELOP*CHANNEL
                         PERSONNEL*ACC
                         PERSONNEL*CHANNEL
BLOCK-CONVERT*MD         DEVELOP*BLOCK-CONVERT
                         PERSONNEL*BLOCK-CONVERT .
ENG*BP                   DEVELOP*ENG. BP
                         SP*ENGBP
ENG*MD                   SYSPROG*EMD
PERSONNEL*NAMES          GAMES*NAMES
                         OPERATIONS*NAMES
                         SHIPPING*NAMES
PROCLIB*MD               DEVELOP*PROCLIB
                         PERSONNEL*PROCLIB
                         SEMA4*PROCLIB
PROGS*AP                 SALLY*AP
                         USERS*PAYABLES
PROGS*PO                 SALLY*PO
                         USERS*PURCH
PROGS*SO                 SALLY*SO
                         USERS*SO
PROGS*VM                 USERS*VENDOR. MASTER
SYSPROG*BATCH            GAMES*BATCH
SYSPROG*BP               GAMES*BP
SYSPROG*ERRMSG           GAMES*ERRMSG
                         SEMA4*ERRMSG
                         STANDBY*ERRMSG
SYSPROG*MD               DEVELOP*SPMD
SYSPROG*STAT-FILE        DEVELOP*STAT-FILE
                         SEMA4*STAT-FILE
SYSPROG*SYSPROG-PL       DEVELOP*SPL
                         GAMES*SPL
SYSPROG*TSYM             SEMA4*TSYM
SYSTEM*ACC               OPERATIONS*CHANNEL
                         SHIPPING*CHANNEL
SYSTEM*MD                SEMA4*SYSTEM
SYSTEM*POINTER-FILE      DEVELOP*PF
SYSTEM*PROCLIB           GAMES*PROCLIB
```

Figure 1: Typical output generated by the LIST.POINTERS proc.

```
    LIST.POINTERS
001 PQN
002 IBH%1:U50BB:
003 IBH%1:G1 1:
004 IF A # SYSPROG]SP X You must
    be SYSPROG or SP!
005 F-OPEN 1 QREFS
006 X Can't F-OPEN QREFS!
007 H CLEAR-FILE (DATA QREFS)
008 P
009 HSSELECT DICT SYSTEM WITH *A1
     "D" AND WITH *A14
010 STON
011 H PQ-SELECT 1
012 P
013 100 MV %1 !1
014 IF # %1 GO 300
015 IF %1 = SYSPROG]SP MV %1 "MD"
016 F-OPEN 2 %1
017 X F-OPEN failed!
018 H SSELECT DICT
019 A
020 IF %1 = MD MV %1 "SYSPROG"
021 H WITH *A1 "Q"
022 STON
023 H PQ-SELECT 2
024 P
025 200 MV %2 !2
026 IF # %2 GO 100
027 F-READ 2 %2
028 X F-READ failed!
029 IF &2.3 = "" MV &2.3 "MD"
030 MV %3 &2.2*"*"*&2.3
031 F-READ 1 %3
032 Continue
033 MV %4 %1*"*"*%2
034 MVA &1.1 %4
035 MV &1.0 %3
036 F-WRITE 1
037 GO 200
038 300 H SORT QREFS ID-SUPP
039 P
```

Figure 2: The LIST.POINTERS proc.

```
     1
001  S
002  0
003  ACCOUNT*FILE
004
005
006
007
008
009  T
010  20

     2
001  S
002  1
003  POINTED TO BY ACCOUNT*FILE
004
005
006
007
008
009  T
010  30
```

Figure 3: Dictionary items for the QREFS file.

then line 14 causes a jump to the final sort of the QREFS file to list the results. If another account name is successfully retrieved, line 15 replaces the account name with the string "MD" if the name happens to match the account currently executing LIST.POINTERS. (A possible improvement to LIST.POINTERS would be to recode lines 15 and 20 to automatically use the name of the account executing LIST.POINTERS, so that lines 2 through 4 could be dropped and LIST.POINTERS could execute unchanged from any account with access to the SYSTEM file, to the QREFS file, and to the Master Dictionaries of the various accounts being searched for Q-pointers.)

When line 16 is reached, the first primary input buffer parameter (%1) is either "MD" or the name of an account. Line 16 opens that name to file buffer 2 so that Q-pointers in the Master Dictionary of any account can later be read at line 27. This means that the Master Dictionary of the account executing LIST.POINTERS (usually SP or SYSPROG) must contain a Q-pointer to the Master Dictionary of every account defined in the SYSTEM file, which is usually the case since the CREATE-ACCOUNT proc will automatically create such a Q-pointer. However, verbs such as ACCOUNT-RESTORE do not create a Q-pointer, so an installation should make sure every account in the SYSTEM file has a corresponding Q-pointer in SYSPROG's Master Dictionary before trying to execute LIST.POINTERS.

Line 17 is the usual error abort for the F-OPEN, and line 20 restores the name of the executing account if it was detected by line 15.

Now that the name of an account's Master Dictionary has been identified, lines 18 through 24 find all Q-pointers in the account's Master Dictionary and save them in select register 2. Note that the SSELECT might abort if the account's Master Dictionary is protected by retrieval codes that cannot be matched by the account that is executing LIST.POINTERS. The analysis of each Q-pointer begins with the retrieval of the next Q-pointer name from select register 2 in line 25. If the list of Q-pointer names is exhausted, then line 26 causes a jump to continue processing the next account. Otherwise, line 27 reads the actual Q-pointer data (with line 28 aborting if the pointer happens to have disappeared since the select). If the file name in the Q-pointer's third attribute is null to denote a default (the Master Dictionary), then line 29 fills the attribute with "MD" to insure a more clear report.

Line 30 creates a QREFS item identifier from the account and file names found in the Q-pointer, and line 31 reads the corresponding cross-reference item, if any, from the QREFS file. If no item is found in QREFS, it simply means that this is the first reference found, and the comment in line 32 allows the failure of the F-READ to be ignored. Line 33 creates the string that names the Q-pointer, and line 34 includes the string in the list of references in the QREFS item just read. Lines 35 and 36 restore the QREFS item identifier and write the record back to the QREFS file. Line 37 repeats the loop for the next Q-pointer.

Once line 38 is reached, QREFS contains all the completed cross-reference items, and a simple SORT produces the final report. Figure 3 shows the dictionary items needed to produce the two report columns.

Whether to help document a complicated system of accounts, or to identify leftover and unnecessary Q-pointers to nonexistent files, or to insure a file is consistently accessed via Q-pointers all with the same name, LIST.POINTERS is a simple but useful and fast-executing utility to do the job. P

SYSMAP

Part 3: Remaining File Input

This third article in a series on the use of cross-references presents the remaining input programs for maintaining the SYSMAP files.

In the last installment, *Part 2: File Format Input* (Pragma #2, page 14), the input program for SYSMAP's FF file was presented. The five screen definitions below (which are shown in I-DUMP format) and the five corresponding programs on page 16 are the remaining programs for maintaining the XB, XD, XF, XP and XT files, which were originally documented along with the FF file in *Part 1: A Cross-Reference System* (Pragma #1, page 22). All of the programs are very similar in design, and provide no more than simple attribute creation and editing capabilities.

In the next installment, a complete set of dictionary definitions, procs and sample reports will be presented, demonstrating how relations captured with these input programs (or even captured by more automatic cross-reference generators) can be output as meaningful and informative cross-reference listings.

```
GET.XB^^^PROG]PROCS]OK^^Create or change Basic prog xref data\\\\
\\\                    Program:\\ PROCs that use program:^0,0^^^S]V]D
^]A^L]L^1]1^Y]]Y^^^7,25]9,25^20]40^^^20]40^^Program:]PROCs that u
se program:]File (FI) or exit (EX)?^22,0]22,0]22,0^22,8]22,23]22,
23^72]57]57^^20]40]2^^^^^F]](´FI´)F\GOK^^^^^^^]PROCS^^(´EX´)E^

GET.XD^^^FILE]WORD]PROCS]WORDS]OK^^Create or change dict xref dat
a\\\\\\\                Dictionary:\\                    Word:\\    P
ROCs that use word:\\     Words that use word:^0,0^]]WORDS^^S]S]V]
V]D^]]A]A^L]L]L]L^1]2]1]2^Y]]]]Y^^^7,25]9,25]11,25]13,25^40]40]40
]40^^^40]40]40]40^^Dictionary (file name):]Word in dictionary:]PR
OCs that use word:]Words that use word:]File (FI) or exit (EX)?^2
2,0]22,0]22,0]22,0]22,0^22,23]22,19]22,20]22,20]22,23^57]61]60]60
]57^^40]40]40]40]2^^^^^^]F]]](´FI´)F\GOK^^^^^^^]]PROCS]WORDS^^(´EX´
)E^

GET.XF^^^FILE]ATR]RPROG]UPROG]RPROC]OK^^Create or change file xre
f data\\\\                File:\\                Attribute:\\    Progr
ams that read:\\ Programs that update:\\ DICT words that read:^0,
0^]]UPROG\RPROC^^S]S]V]V]V]D^]]A]A]A^L]L]L]L]L^1]2]1]2]3^Y]Y]]]]Y
^^^3,23]5,23]7,23]9,23]11,23^40]40]40]40]40^^^40]40]40]40]40^^Fil
e name:]Attribute name:]Programs that read:]Programs that update:
]DICT words that read:]File (FI) or exit (EX)?^22,0]22,0]22,0]22,
0]22,0]22,0^22,10]22,15]22,19]22,21]22,21]22,23^70]65]61]59]69]57
^^40]40]40]40]40]2^^^^^^]F]]]](´FI´)F\GOK^^^^^^^]]RPROG]UPROG]RPROC
^^(´EX´)E^

GET.XP^^^PROC]PROCS]OK^^Create or change proc xref data\\\\\\\
                PROC:\\            Calling PROCs:^0,0^^^S]V]D^]A^L]
L^1]1^Y]]Y^^^7,25]9,25^40]40^^^40]40^^PROC:]PROCs that execute th
is PROC:]File (FI) or exit (EX)?^22,0]22,0]22,0^22,5]22,29]22,23^
75]51]57^^40]40]2^^^^^^F]](´FI´)F\GOK^^^^^^^]PROCS^^(´EX´)E^

GET.XT^^^FILE]ATR]PROGS]PROCS]STEPS]DW]OK^^Create or change trans
late xref data\\\\\                File:\\
Attribute:\\   Programs that translate:\\      PROCs that transla
te:\\     Screens that translate:\\ DICT*Words that translate:^0,0
^]]PROCS\STEPS\DW^^S]S]V]V]V]V]D^]]A]A]A]A^L]L]L]L]L]L^1]2]1]2]3]
4^Y]]]]]]Y^^^5,28]7,28]9,28]11,28]13,28]15,28^40]40]40]40]40]40^^
^40]40]40]40]40]40^^File name:]Attribute name:]Programs that tran
slate:]Procs that read/write/translate:]Steps that translate:]DIC
T*Words that translate:]File (FI) or exit (EX)?^22,0]22,0]22,0]22
,0]22,0]22,0^22,10]22,15]22,24]22,32]22,21]22,26]22,23^70]65
]56]48]59]54]57^^40]40]40]40]40]40]2^^^^^^F]]]]](´FI´)F\GOK^^^^^^
]]PROGS]PROCS]STEPS]DW^^(´EX´)E^
```

SYSMAP Input Screens

```
    GET.XB
001 OPEN "DF" ELSE STOP "SYSMAP1"
002 READ SCREEN FROM "#GET.XB" ELSE STOP "SYSMAP2"
003 OPEN "XB" ELSE STOP "SYSMAP3"
004 100 PRINT CHAR(12):
005 200 INPUT ITEM USING SCREEN, "" SETTING NEXT.STEP ELSE STOP
006 PROG = ITEM<1>
007 READ ITEM FROM PROG ELSE ITEM = ""
008 INPUT ITEM USING SCREEN, ITEM AT NEXT.STEP ELSE GO TO 100
009 WRITE ITEM ON PROG
010 GO TO 100
011 END

    GET.XD
001 OPEN "DF" ELSE STOP "SYSMAP1"
002 READ SCREEN FROM "#GET.XD" ELSE STOP "SYSMAP2"
003 OPEN "XD" ELSE STOP "SYSMAP3"
004 PREV.FILE.ATR = ""
005 100 PRINT CHAR(12):
006 200 INPUT ITEM USING SCREEN,PREV.FILE.ATR SETTING NEXT.STEP ELSE STOP
007 FILE.ATR = ITEM<1>
008 PREV.FILE.ATR = FILE.ATR
009 IF FILE.ATR = "" THEN GO TO 200
010 ATR = ITEM<2>
011 IF ATR # "" THEN FILE.ATR = FILE.ATR:"*":ATR
012 READ ITEM FROM FILE.ATR ELSE ITEM = ""
013 INPUT ITEM USING SCREEN, ITEM AT NEXT.STEP ELSE GO TO 100
014 WRITE ITEM ON FILE.ATR
015 GO TO 100
016 END

    GET.XF
001 OPEN "DF" ELSE STOP "SYSMAP1"
002 READ SCREEN FROM "#GET.XF" ELSE STOP "SYSMAP2"
003 OPEN "XF" ELSE STOP "SYSMAP3"
004 PREV.FILE.ATR = ""
005 100 PRINT CHAR(12):
006 200 INPUT ITEM USING SCREEN,PREV.FILE.ATR SETTING NEXT.STEP ELSE STOP
007 FILE.ATR = ITEM<1>
008 PREV.FILE.ATR = FILE.ATR
009 FILE.ATR = FILE.ATR:"*":ITEM<2>
010 READ ITEM FROM FILE.ATR ELSE ITEM = ""
011 INPUT ITEM USING SCREEN, ITEM AT NEXT.STEP ELSE GO TO 100
012 WRITE ITEM ON FILE.ATR
013 GO TO 100
014 END

    GET.XP
001 OPEN "DF" ELSE STOP "SYSMAP1"
002 READ SCREEN FROM "#GET.XP" ELSE STOP "SYSMAP2"
003 OPEN "XP" ELSE STOP "SYSMAP3"
004 100 PRINT CHAR(12):
005 200 INPUT ITEM USING SCREEN, "" SETTING NEXT.STEP ELSE STOP
006 PROC = ITEM<1>
007 READ ITEM FROM PROC ELSE ITEM = ""
008 INPUT ITEM USING SCREEN, ITEM AT NEXT.STEP ELSE GO TO 100
009 WRITE ITEM ON PROC
010 GO TO 100
011 END

    GET.XT
001 OPEN "DF" ELSE STOP "SYSMAP1"
002 READ SCREEN FROM "#GET.XT" ELSE STOP "SYSMAP2"
003 OPEN "XT" ELSE STOP "SYSMAP3"
004 100 PRINT CHAR(12):
005 200 INPUT ITEM USING SCREEN, "" SETTING NEXT.STEP ELSE STOP
006 FILE.ATR = ITEM<1>
007 ATR = ITEM<2>
008 IF ATR # "" THEN FILE.ATR = FILE.ATR:"*":ATR
009 READ ITEM FROM FILE.ATR ELSE ITEM = ""
010 INPUT ITEM USING SCREEN, ITEM AT NEXT.STEP ELSE GO TO 100
011 WRITE ITEM ON FILE.ATR
012 GO TO 100
013 END                                                          Ⓟ
```

SYSMAP Input Programs

user profile

Are all data processing installations the same? How do managers actually manage, how do programmers program, how do operators operate, how do users use their data processing systems? What defines the leading edge of current, modern information processing?

In this regular department, Pragma will be interviewing personnel at a variety of installations, to reveal the who, what, when, where, why and how of actual data processing organizations.

Rainbow Natural Foods is located in Denver, Colorado and employs about 100 people. For this issue's user profile, Pragma interviewed Stephen R. Kowarsky, Rainbow's Director of Information Systems. The company is a wholesaler and retailer of natural foods.

Pragma: Give us some of Rainbow's history.

Kowarsky: Rainbow started as a retail store six or seven years ago. It began wholesaling because it developed good lines of supply, good prices, and so on. We are, at this point, the largest natural foods wholesaler in the Rocky Mountain area. Customers include natural foods retailers, food co-ops, manufacturers, restaurants, institutions of various kinds. We have moved and expanded the warehouse a number of times in the last several years. We now have over 30,000 feet of warehouse space, and we ship as far east as Chicago. We do a lot of business in the Middle West, and of course a lot of business in the Rocky Mountain area and the western slopes.

Pragma: How did Rainbow data processing get started?

Kowarsky: In March of 1979 Rainbow hired me to automate their systems. At that time, the wholesale business was just beginning to boom. It was about a year and a half old at that point. Everything in the company was manual. The only machines in the company were a cash register and an adding machine. The management of the company recognized that, in order for particularly the wholesale end of the business to grow, they would have to automate their systems. That is, if they were going to be able to grow without adding huge numbers of people. Growth in the number of products carried, growth in the number of customers served, growth in the size of the orders to those customers, growth in the size of the warehouse. All those kinds of growth were clearly anticipated, and they knew that they were going to have to automate in order to accomplish that. Some of the top management at Rainbow were friends of mine and they called me and said "How would you like to come and do this?" It sounded like fun, so I said that I would, and I did.

Pragma: How did you do when you arrived?

Kowarsky: At the time that I came, I took a very open minded approach. It had been a while since I had really investigated the state of the art in minicomputer systems. I knew a lot was going on. I did a lot of research. I told them that I expected to do six months to a year of research before they would see much in the way of results. It turned out that it was considerably less than that, but that was acceptable to them, because they wanted to do it right the first time. I might add that I think that they went about this in the right way. I talk to people that are in a position similar to the one Rainbow was in four years ago, and I think Rainbow did it very intelligently. They saw that they needed to automate enough in advance, before the need was really critical, that they didn't have to try to rush to get something happening, quick and dirty. So that's one thing that they did right. The other thing that they did right was they brought in a fairly high level professional person to work on it from the beginning. I'm speaking of myself. So I had an opportunity to learn the business. One of the first things I did when I came was do many of the jobs, many of the clerical jobs that people were doing at that time. There was a fairly immediate need in accounts payable, because they were manually writing hundreds of checks a month, and I actually did all the accounts payable for a couple of weeks, using the manual system. It turned out that the first thing that we automated was accounts payable and general ledger. Even though everybody knew that the big thing was going to be order entry and inventory control for the distribution operation, we put that off for awhile and did accounts payable and general ledger first, because it involved a lot fewer people. The impact on the organization was a lot smaller, so it was a chance for the organization to get exposed to data processing without going through the kinds of traumas that you go through when you affect dozens of people and dozens of systems, which is what happens when you automate order entry and inventory.

Pragma: What was the result of your research?

Kowarsky: I was looking at DEC and I was looking at Data General and I was looking at Hewlett Packard, and simultaneously I was looking at software and distribution packages and that kind of thing. I had one friend who worked for a company here in Denver named Gathers Software, and they had a timesharing program going. That was of interest to us because we were feeling very tentative at first about everything that we did. There was not even a foregone conclusion that we would have our own system, or timeshare, or use a batch service bureau or whatever. Gathers was on a Microdata system. I looked at their software, and it looked reasonably good, and the terms were attractive. No money up front, no long term commitment, no big investment in equipment. They rented the equipment to us. Basically he made us an offer we couldn't refuse. We did go into a timesharing arrangement for accounts payable and general ledger, and that, as it turned out, was the decisive step, probably more decisive than we realized at the time. I think that's a clever and a very valid marketing technique: to offer a client an easy entry into a timesharing environment. If it's a good environment, the client does get hooked. Once you've gone through the process of converting your systems to a particular set of software, you're reluctant to turn around and back out of it and do it some other way. That was definitely a turning point in our development. I was not at all knowledgeable at that time in the Pick system. Most of my work was with large scale IBM systems. But we got on this timesharing system, and I started playing with it, and began to learn about it, read the manuals, and do a lot of things with it. There was another person at the company too who was the controller, who also became very interested in it. He didn't have a computer background at all, but he had a very strong aptitude for it and a very keen interest in it, and so he and I got involved in it. Of course he had to be very involved in the project anyway as the controller. He was the other main person involved in figuring out what these systems were going to do and how we were going to set them up. He also, as it turned out, became very involved in the technical aspects of data processing. Between the two of us, over the course of three or four or five months, we became a fairly high powered team in Pick systems, to the point where we had the confidence that if we went and bought our own system, we'd be able to support it. Our timesharing costs were getting to the point where it looked like we ought to buy our own system. Plus that was still before we did anything with order entry and inventory control, which obviously would have cost a lot more in terms of timesharing. So we went out and we bought a used core Microdata machine. We got that in December of 1979.

Pragma: When you were timesharing off the Gathers system, what was your configuration?

Kowarsky: All we had was one terminal and a slave printer, operating at 1200 baud. It was used for all payables data entry, for all check writing, all other general ledger journal type entries, and for a variety of reports.

Pragma: What was the size of the company when you joined to start the automation effort?

Kowarsky: They were in a warehouse one half as big as it is now. It hadn't been racked yet. You know, modern warehousing pallet racks. Loads are palletized and stored on racks using a forklift. Primitive warehousing just piles everything up on the floor, and on top of each other when necessary. One of the first signs of a warehouse's move into a more sophisticated environment is when it's racked. The dollar volume of the business was a fourth of what it is now. The number of employees at about forty percent. We've quadrupled the business with just about a doubling of the employees. That's the leverage of the automation.

Pragma: Were you searching for software?

Kowarsky: I was researching distribution systems. Because we really liked the Pick system, I was researching them for the Pick. I came across a couple. The one that sounded the most interesting was a system out of Seattle. I had some preliminary phone conversations with them in around September of '79. They told me that they'd just rewritten their package. I decided to take a quick trip up there to see a demo of the system. What I found when I got there was a system that looked like it had a lot of good fundamental thinking in it, but was far from a finished product. It had a lot of big and small problems that were pretty clear, from even a slightly more than superficial delving into it. I had prepared a whole questionnaire of features, and I had gone through this questionnaire with several people on the phone and it met a lot of the criteria. It had a nice overall feel to it somehow, but it really had a lot of weak points also. I made a deal that I would stay there and help them get the package ready. You know, smooth the rough edges, fix the bugs and overall make it a really marketable package, in exchange for acquiring the package for Rainbow. Now what was in it for Rainbow and for me personally was I got to do a lot of work in an environment where there were a lot of experienced Pick people. I still really hadn't been all that up to speed on the system at that time. It's always been one of my adages that the best way to learn something is to put yourself in the position where you have to produce it, and then you learn whatever you need to know in order to produce the thing that you have to produce. That's definitely been the way my own knowledge of data processing has evolved, and that's always the advice I give to people. I was kind of sticking my neck out. You sort of act confident, and you impress them with your knowledge of the application, which I did at that point. From my experience at Rainbow I had a pretty good feel for the distribution application. As far as my ability to program, I just had confidence that if I threw myself into the situation, I'd be able to do it. You have to overcome a certain amount of initial fear. But anyway, they took me up on the offer, and what I thought was going to be a two day trip turned into a five or six week stay. I was able to further my own education in the Pick world a great deal. I got to learn the system very thoroughly in advance of the system arriving at our doorstep, from people who knew something about it. I also got to adjust some of the features in ways that were particularly congenial to our needs. I didn't do anything ridiculous to it. I didn't bend it, but in any application there are many, many options and ways that things can be done. You can't really say that one is more valid than the other. Some apply to different types of industries, some are matters of choice, some have tradeoffs. I felt that I did some very good things with the package, and some of them were definitely oriented toward needs that I knew we had. The actual deal was they were going to pay for my time and we would pay for the software, but it was pretty much a wash. Really, we didn't even commit to buying it when I committed to doing this. I figured if worst came to worst it would give me an in-depth evaluation of the thing, at their expense rather than at my expense. We did buy it, and incidentally, subsequently thoroughly rewrote it all over again on our own. Completely restructured the data base and everything.

Pragma: So you acquired the software at the same time as your hardware?

Kowarsky: Shortly thereafter. They probably were virtually simultaneous. I was in Seattle in October. We were in the process of making a hardware decision right at the same time. I was constantly on the phone back to Denver to the controller. I came back, we ordered the software, we ordered the hardware. It pretty much all happened at the same time.

Pragma: How did you decide on a used Microdata and its

initial configuration?

Kowarsky: We always try to be cost conscious, and we thought we could save money with used hardware. There was a particular machine available and we thought that it could do the job. The configuration was 64K of memory, 8 ports, two 10 megabyte disk drives, an 800 BPI tape drive. We had that machine starting in December of '79. We went live with full blown order entry and inventory in May of '80. We were already live with general ledger and accounts payable software that we'd brought over from Gathers. It didn't take too long before we really started pushing the limits of that configuration on all fronts. We added 32 more K of memory. At one point we actually stuck more ports on there temporarily although we were pushing the limits of the number of ports we could maintain acceptable response time on. Twenty megabytes of disk sounded like a lot coming out of the timesharing environment. It still to this day amazes me how much we got out of that 20 megabytes. All the accounting applications, and the full order entry and inventory. But it was getting to the point where you'd have to purge the thing out to tape every weekend just to maintain any kind of disk space at all. You sort of lived in constant fear that somebody wouldn't be able to log on because there wouldn't be enough workspace.

Pragma: How well did the Gathers software fit into your distribution package?

Kowarsky: We integrated the two, and it was not a difficult integration. Basically we took the data from order entry every night and spun off a sales journal in the format of the Gathers general ledger. We didn't try, and still haven't tried, to integrate purchasing and accounts payable.

Pragma: What happened when you outgrew your configuration?

Kowarsky: There's always the battle of the data processing person against the president of the company, trying to get more money spent on more hardware. I jumped up and down and wrote memos and did everything I could. My concern was, and still is, that we are a very response time critical environment. We take the orders in real time over the phone into the terminal, and there are a number of other situations that are response time critical as well. If the response time isn't there, you not only frustrate the customer, but you frustrate your own people tremendously. Particularly the situation of taking an order, trying to deal with a customer. If the response time isn't there in the terminal, it's just an extremely stressful situation for the user. If your system puts the user under stress, everything falls apart. They're going to hate it and they're not going to want to work with it. I got a classic comment from one of the people working on the front desk before we really automated. She is still here and is now very much a fan of the system. But I had said something to her about talking to the computer and she said "I'd rather talk to a bag of beans!" I've always thought that was a classic user resistance comment. Anyway, I always wanted and still always want the system to function really optimally. For me, that means response times of a half second or less. If it's more than that in the kind of environment that we're in, it starts creating real frustration. Plus, as you become more sophisticated in data processing, you want to do more analysis of your data. You want to get more information out of that data, and that requires disk capacity to keep some history of the system so that you can analyze it. I was selling my pitch to the managers, and I finally convinced them that we should do something. Microdata announced the 8000, and I said an 8000 is what we should do. I had written a proposal and done a financial analysis and all that and submitted it. The answer that came back was wait, for financial reasons primarily. Some time went by. I tried to keep in touch with people. I was beginning to develop some national contacts, getting to know people who were actively involved in the Pick system. Everybody was telling me that R80 was the way to go, that Microdata just didn't have it. Everybody was saying that R80 is much more efficient and has a lot of nice things to it, and you should at least convert your Microdata to an Evolution and all that. Enough people were telling me that, that I felt like I at least had to consider it. To me the tradeoff was that I didn't want a conversion. Even if it was a mild conversion, I didn't want to touch it, so I was tremendously biased toward the Microdata 8000. But finally I decided I'd better look into it. I would say that I made a decision that I was going to have to seriously check out the Pick alternative in February or March of 1981.

Pragma: For a lot of our readers who are strictly Microdata users, can you define R80?

Kowarsky: Dick Pick left Microdata and started supporting the Pick operating system in a separate and parallel path, on his own. First with Evolution then with Pick Computer Works. R80 was the current version of the Pick system at that time. In fact, it basically still is. The "R" stands for release. He had release 78, then 79, then release 80. I think they roughly correspond to years, but R80 probably came out toward the end of 1980. R80 was the then current version of the Pick implementation of the Pick operating system.

Pragma: What did you do to investigate R80?

Kowarsky: I decided that if I'm going to consider the alternatives, I should consider them very carefully. So I wrote a set of benchmarks. I tried to measure as many things as I could about the machine. I arranged to run them on a Microdata 8000, a Mentor and an Ultimate. I ran the benchmarks locally on an 8000, and we went to Ultimate and to ADDS, in New Jersey and Long Island, to run them on those machines. Both companies were incredibly cooperative, turning over a system to us for two solid days, so that we could run these things. I learned that both the Mentor and the Ultimate outperformed the 8000 substantially, particularly in terms of their ability to maintain response time under a heavy processing load. The Mentor, at half the price of the 8000, was faster, and it was closer to the Ultimate than it was to the 8000, although the Ultimate was the superior of the three. So what we had to evaluate were two things: the price/performance, which was screaming at you, and the risk, which was a kind of contrary factor, because the Mentor at that time was an unproven product. There really weren't any of them up and running in the field, whereas the Ultimate was proven. The benchmark more or less ruled out the 8000. Ultimate cost the same as the 8000 and maybe a little more, but way outperformed it. We tossed that around for quite awhile, and we finally decided to take the risk.

Pragma: What about other risks, like conversion of software?

Kowarsky: Yes, that was a risk factor too, there's no question about it. At that point, I had investigated the differences enough to feel reasonably confident. We figured that if worst came to worst and the thing didn't work, at least the conversion work that we did for the Mentor would be able to turn around and apply directly to the Ultimate, since the two systems are both basically versions of Pick's R80. So we decided to go with the Mentor. "We" is the executive committee of the company. The President, the General Manager, the Controller and I.

Pragma: Was it essentially them approving your recommendations?

Kowarsky: To be perfectly honest, I was recommending Ultimate. I didn't like the risk of an unproven product.

Pragma: But they changed your mind because of the cost?

Kowarsky: Yes. I think their position was that this was an offer that we can't refuse, and if it doesn't work out we can always spend the additional money and get the Ultimate. We tried to structure the deal so that we could get out of it, if the machine didn't perform at all. I was probably more aware than the other people were of the kinds of horror stories that have happened with new computers that have been put on the market and totally bombed, then called back off the market and take another two years before they were released. It's happened more than once. I just felt that it was an unwise risk. I was very shaky about it. My initial recommendation was the safe one for the Ultimate. But we looked at the facts together, and we did come to a decision together. It was a group decision. So we ordered Mentor. We got it in July of '81, and we took our time. We just played with it for a month. We started serious conversion in August. In September we went live with the order entry and inventory side of things, and we kept accounting on the Microdata. In November we went live with accounting on the Mentor. The conversion was...I guess all I can say is that I was genuinely surprised at how easy it was.

Pragma: Why was the conversion easy? Because R80 so closely duplicates Microdata's features?

Kowarsky: The thing that most surprised me was DATA/BASIC. The two BASICs are very similar. Most of the differences can be automatically converted by a program, and a program was supplied. I ran my source code through this program and it changes the LOCATE statement, which is different. If you used any of the INS and DEL statements, it changes those back, although the Pick does support the new format of EXTRACT and REPLACE. There are a few other oddities. It's not "MD", it's "MR" for the conversion code. A couple of things like that, nothing major. I ran a huge source library through this conversion program, compiled the new programs, and most of the programs functioned right off the bat. I was just really delightfully surprised. ENGLISH is very similar. Again there are differences in the dictionaries that are taken care of by an automated conversion program that's supplied. There were a few things that had to be tinkered with. ENGLISH has a lot of its own funny quirks when you get into multivalues, taking the function of the sums rather than the sum of the functions, and things like that. Most of it worked very well. It's the kind of thing where if you know ENGLISH, you run the report on the new system and, if it doesn't work right, just go play with it a little, just like you play with ENGLISH to make it do anything, and you'll get it working. There was one major capacity that was lacking in R80 which was IF-THEN-ELSE in the A-correlative. We made extensive use of that on the Microdata, because it's a very powerful feature. That was a hard one at first. We took other ways around it for some items. Ultimately what we did was actually add IF-THEN-ELSE to the Mentor, and it's the only Pick implementation that has it, outside of Microdata.

Pragma: What about procs?

Kowarsky: Procs are the biggest difference, because there is no such thing as a PQN proc in Pick land. PQN was added by Microdata after Pick left. Apparently it's based on stuff that was done at SMI. PQN proc is kind of an SMI product instead of a Pick product. Pick never put it in. So if you have taken advantage of F-READs and ampersands and percent signs and the MV command and all these wonderful things that are features of PQN proc, you have some truly non-trivial conversion effort to make in converting your procs. That is a hand conversion. The proc on R80 is more powerful than Microdata's PQ proc, but it's by no means anything like PQN proc. There is no F-READ, there is no MV, there is no %2.

Pragma: So how much of a problem was it for you?

Kowarsky: Well, it was a week's work. You sit down and you do it.

Pragma: Just how compatible do you think different Pick systems are going to remain in the future?

Kowarsky: That's a really good question. I think that there are some contrary forces acting in the marketplace. The vendors who have done Pick implementations have an incentive to set themselves apart from the other implementations by adding this feature and that feature. Pick has or maybe should have an incentive for the Pick implementations to be as similar as possible, so that there can be transportability, which is supposedly one of the selling points of the system. I think it's going to be very interesting to see which one of those forces wins out, or just how it develops. I know some people who are taking the philosophy of not using anything that is not transportable, that they can't pick up and put on another system and run it as is. I talked to this one dealer who sells Microdatas and Ultimates and Mentors. He won't write anything except the lowest common denominator. So they don't use the LOCATE statement, they code it out. They don't do anything with PQN proc, they don't use PROCREAD and PROCWRITE. They don't use IF-THEN-ELSE. That's one alternative, but that's not a very attractive alternative, because what are those tools there for? They are there to be used. It's good to use them. I think the course that we've chosen is to make a commitment to one particular type of implementation, and to more or less plan on sticking with that vendor unless there's some extremely compelling reason not to. One of the things that I did try to establish was that ADDS as a company was really committed to this product. That they were really serious about it, and that they saw a future in it. When we went to ADDS, we talked extensively with them on this point. We met a number of the people on the development team. We tried to satisfy ourselves that they were people who were going to stand behind what they did and were capable of doing what they were trying to do. Since then we've more or less put ourselves behind ADDS 100%. We give them all the feedback we can about what we'd like to see. When we've developed things that we think would be helpful to them, we've talked to them about it, and some of those things have been incorporated into the system, like IF-THEN-ELSE for example. For screen processing, ADDS has incorporated a version of WIZARD. It really

Pragma is Preparing an Index

Pragma will be periodically publishing a complete master index for all articles appearing in **Pragma**, as a reference tool for its readers. Articles will be cross-referenced by keywords found in each article's title, by the author's last name (for submittals), by additional miscellaneous keywords for various subject areas, and by department. Also, all advertisements will be indexed by vendor, product name and product type.

isn't WIZARD. It's based on WIZARD, but it's an expanded and enhanced version of it, which I like and I use.

Pragma: What's your evaluation of that product?

Kowarsky: You're going to get a very biased opinion. I helped ADDS evaluate a number of the alternatives, and when they decided on WIZARD, they asked me to turn it into what they have, which they call Implementor, so I'm really the author of Implementor.

Pragma: What was your initial Mentor configuration?

Kowarsky: My initial Mentor configuration was 384K MOS memory. It comes in 128K per board. The Mentor 4000 has a capacity of four boards and we started with three. We got 16 ports. We started with one 30 megabyte drive, but the order was for one 60 megabyte drive and the 60s weren't available so they gave us a temporary 30 megabyte drive. It uses the Cipher Microstreamer drive. It's industry standard half-inch 9-track 1600 BPI tape. It has a streaming mode, and it has a start-stop mode as well. The reason they call it the Microstreamer is because it's a little thing and the tape lies on its side. It's got a door that opens like a pizza oven, and you just shove the tape in and close the door and it's self-loading.

Pragma: When you did the benchmarks and you found such a difference in speed, what did you attribute that to?

Kowarsky: The configurations were as much as possible equalized, so I attributed it to the monitor itself — the very lowest level control processors that schedule process activation and priority just being a lot better in the R80 implementation. As I understand it, it's one of the things that's most different from the version that Microdata has. Some of the benchmarks I ran were strictly CPU bound. In fact, one of them was a dynamic array manipulator which did nothing but insert and replace and build this huge dynamic array. It was much faster on Mentor than either of the other two. The Z8000 processor is very fast.

Pragma: Considering Rainbow's success to date, do you see problems with keeping your software working as Rainbow grows and expands?

Kowarsky: The systems are multi-company and multi-warehouse. They are designed to allow a lot of flexibility, but what's really going on in data processing here is a growing maturity of the other managers in terms of utilizing the kind of information that you can get out of a system like this. This is probably the slowest thing to come. Everybody can see that if you sort the picklist you can pull the order faster, and that if the computer prices, extends and totals the invoice, it can do it faster and more accurately than a person. But for people to really understand and use more and more sophisticated sales analysis, financial analysis, return on investment analysis kinds of tools — all of which are potentially available in the data base — that's what takes time, and that's where most of my energy goes right now.

Pragma: What are your preferences regarding software development? Packages versus custom programming, programming in-house versus using consultants...

Kowarsky: I'm glad that we started out with packages, even though in the case of our distribution system, it's been almost a total rewrite. There is hardly a line of code that hasn't been changed, hardly a data structure that hasn't been changed. The general ledger, accounts payable, accounts receivable and payroll have all been modified. Some more, some less, but not to the same extent as order entry and inventory. But I would say that I'm glad that we started with packages. And the more you've got someone who knows in real depth what's going on in the computer, and who has the time and motivation to know in real depth what the needs of the organization are, the more you're going to get out of it. It's that simple. We never needed to resort to contract work or consultants. If someone didn't have expertise available in-house, that would be the way to go. But an employee of a company who has a long term commitment to the company just has a lot more incentive to do the kind of a job that is in the best interests of the company. I'm not trying to put down consultants. But I do advise people that are getting started to hire a data processing professional on their staff. I think that it's worth it.

Pragma: If you were to design a system from scratch today, what would be your preference for a machine?

Kowarsky: It would be between the Mentor and the Sequel. At this point, we've put a lot into Mentor ourselves and it does a lot of good things, and I'm very intimate with it now. I don't feel as intimate with a Microdata as I do with the Mentor. But still, if I had an unlimited budget, it would be nice to have a Sequel. The thing that I hate most is my intercom buzzing from the users asking me if there's anything I can do to give them more speed, better response time. That just depresses me tremendously. Partially because sometimes I can't. Partially because if I can, it's at someone else's inconvenience, who's trying to run a report or this or that. And partially because I just don't think it's a completely successful installation if that's going on. If I had a Sequel in here, that would never happen, that's for sure.

Pragma: How much has your configuration changed since you initially got the Mentor?

Kowarsky: Well, we now have two 60 megabyte drives, we now have another 8 ports for a total of 24 and I now have filled out my 512K.

Pragma: How many reels does your backup take?

Kowarsky: I have my system divided into fixed accounts and dynamic accounts, so my nightly backup is one reel, and occasionally it's a reel and a very tiny bit.

Pragma: What are all the applications you actually have running now?

Kowarsky: General ledger, accounts payable, accounts receivable, payroll and bad checks (particularly oriented at the retail store). We have a sales history system which is somewhat standalone, in that it incorporates all the retail data. A full blown order entry, purchasing, receiving, and inventory control. We have word processing. We have bits and pieces of trucking for the transport operation. There's a lot of different pieces. We have an investment cash portfolio system. We're talking about little doodads here and there.

Pragma: Do you make heavy use of record locking and immediate updates, as opposed to logging transactions and batching them for some sort of overnight run?

Kowarsky: My philosophy on that is that I do immediate updates wherever I need the information immediately. And that includes my inventory, my payables, and my general ledger. But there is batch style processing too, particularly for interfaces between systems and for historical and sales analysis type of information. A lot of that is handled in batch. To me it's a tradeoff between immediacy of information and response time. I don't want to be doing a lot of processing during the day when I've got my people taking orders, if I don't need that information.

Pragma: How are your terminals distributed and used?

Kowarsky: Four and a half terminals are used for order taking.

The other half of that terminal is used for trucking. Two terminals are used for purchasing and inventory control and pricing and so on. One terminal is used for the produce department, all of their purchasing and inventory control and pricing. Produce is a little bit of a separate world within the food business because it's the live stuff. Everything else turns once or twice a month, produce turns once or twice a week or more. It's just a whole different world, so there's a terminal for produce. Then in accounting, there are four terminals. There are four people in the accounting department and they all have terminals. Accounts payable has one, accounts receivable has one, the controller's assistant has one, and the controller has one. The front office receptionist has one. She gets a lot of overflow data entry and word processing. I have one for whatever I do. There are two ports that have serial printers on them. They are not serial spoolers, partly because I implemented them before I had a serial spooler available, and partly because I like it better the way I'm doing it. The printers are under the control of a proc in an endless loop that wakes up, reads selected control files and does whatever the record tells it to do. One of those is printing invoices all day, and the other one is out in the warehouse printing picklists. The other terminal is in my house. I have a dial up too.

Pragma: In a conversation we had before this interview, you mentioned your ideas about using standard software resources to do modeling...

Kowarsky: You can create dictionary items that incorporate formulas and procedures very similar to the way that you can put formulas into Visicalc. You don't have quite the cell by cell control that you have in Visicalc, but I've done a lot of very useful modeling using existing data bases as the lines of my report, and creating complex correlatives to model that information in various ways.

Pragma: Do you actually use any modeling packages?

Kowarsky: I have looked at a couple of them. So far, although I like the idea of modeling a lot, I've been a little disappointed in what I was able to do with them. To be very frank, I have some reservations about those packages in the Pick environment. I think that they are best utilized in a dedicated microprocessor environment, where you can have something that's written in machine language, and it just whips through the thing in a second.

Pragma: Does your company use Visicalc?

Kowarsky: No, we don't. We don't have any micros right now. We might get one. But I have done some really nice things just using that technique with ENGLISH and existing databases. If you have a personnel data base in your payroll system, you can do very interesting modeling. Hypothetical salary changes and things like that. You can do very interesting modeling of what would the effect be if you changed the margin by such a percent on this product line, or changed a product mix this way. Just using ENGLISH, I've done some really nice things. I've had a love affair with ENGLISH for years, while the spread sheet systems are still a strange tool, so there's that whole intimacy thing again. When I sit down with ENGLISH, I feel like I'm on home territory. It's nothing to me to write a three or four line IF-THEN-ELSE A-correlative that does really complicated things.

Pragma: How are you handling your backlog of work?

Kowarsky: It really is hard, because a fair amount of my time is spent responding to specific needs, because I am really the only one available. My biggest problem is that to do real development requires uninterrupted concentration, and I'm in a situation where I'm frequently interrupted. As far as prioritizing, I'm in a pretty good position to do that, because as part of our executive committee, I do have a perspective on where we're at. The obvious criteria to apply when there is a priority question is: what's going to have the biggest potential impact on the bottom line of the organization? That's what you try to look at.

Pragma: What's your experience with plug compatible peripherals?

Kowarsky: I've had good experience with printers, for the most part. I've got a Printronix 600 line a minute printer as the system printer, I've got a TI 810 on one of the serial ports, and I've got a GE Terminet 200 on the other serial port, and they're all work horses. I've had basically no problems at all. For terminals I'm using a couple of old Regent 100s, primarily Regent 40s, and I just got in a couple of new Viewpoint 60s. Of course these are all ADDS products, so you expect compatibility and you get it. But I was using both the Regent 100s and the Regent 40s on the Microdata. I never had a Prism, and I had no problem.

Pragma: How is your field service arranged?

Kowarsky: The maintenance contract is with ADDS. When there's a hardware problem, you call an 800 number and you get the...ta daa.. Mentor Response Center. You talk to supposedly an expert who diagnoses your problem and dispatches NCR. The NCR field organization actually handles it. ADDS is a wholly owned subsidiary of NCR. So they plugged into that network, which is a pretty good network.

Pragma: How knowledgeable about Mentors have you found the NCR service people to be?

Kowarsky: I have found the NCR people to be a very high quality people. They're not extremely knowledgeable about Mentors because ADDS has set it up so they're not supposed to be and don't have to be. The idea is there are diagnostics in the machine. When you call the response center, they actually run them remotely for some people. For me, I run them and discuss it over the phone. I think that's quite common, they're not that difficult to do. And somehow together we reach some kind of conclusion as to what's wrong with it and what we're going to do about it. Then NCR comes out and, at least ideally, replaces the third memory board or swaps the disk drive or something like that.

Pragma: How has reliability been?

Kowarsky: Reliability has been excellent. I did go for a period of probably nine months without any kind of a hiccup whatsoever in the system. Then last month I did have a disk problem. I had a couple of crashes, and they came out and swapped a disk. Now it seems that *that* disk is developing a disk problem. I've had a couple of crashes and we're going to swap a disk on Monday. A little bit of "when it rains it pours." Overall, reliability has been excellent, and there has been very little unplanned downtime. Knock wood.

Pragma: What has been your experience when new Mentor software releases are delivered?

Kowarsky: In my opinion, ADDS really doesn't have a release procedure down right. The way it works is to me too complicated. You've got to do a file save and then you do a binary restore of the new release. Then you can't restore all files, because you'll wipe out the new SYSPROG with your old SYSPROG, so you do individual account restores. It's a klugey process, and if it's klugey for me, I can't imagine what it is for an unsophisticated end user. So I always dread it when a new release comes out. I think they really need to work on that. The other thing that they really need, unlike Microdata, is that they do not patch the system in any systematic way.

Occasionally some people will get a patch if it's really badly needed, but only some people. If they really scream for it.

Pragma: How good is your user and programmer documentation?

Kowarsky: I wish I could say we didn't have a documentation problem, but we do. The tendency towards "I understand it, and I'm doing it, and I'm the only one who has to know about it, so why do I have to write it down?" is a very difficult tendency to resist, even when you're aware of it. I've got other things to do besides write it down. My major effort to solve that problem now has been to work with about one third of somebody's time: a person who has been in the inventory control area and has learned a fair amount about the system and who is reasonably comfortable with the Editor, with ENGLISH, with the basic structure of the system. I've been working with this person, trying to bring them along into the whole area of what the systems are all about. As far as documentation, the files are documented, many of the procedures are documented in writing, not all exactly in the same place, or in exactly the same format. But there is a lot that's in my head and it's a problem.

Pragma: What would happen if you were suddenly gone?

Kowarsky: If I was suddenly gone, they would probably call in the other guy who was here for a long time. He would be in a position to tide them over. If I was hit by a truck, it would be a serious problem. For awhile it was hard for me to leave at all. Now I'm at a point where I can take a vacation, take a few days, but I need to leave a number.

Pragma: What do you do to tune the system and improve throughput?

Kowarsky: Several things. On the real response time critical applications, you just go through every single line of code and try to make it run at absolutely maximum efficiency. Very much related to that is to make sure that the files you're accessing on line that are critical to response time are allocated optimally. You've discussed that in Pragma. Another aspect of that is to, as you find it necessary, watch your job mix during the day. Make it as convenient as possible for people to run things overnight. I have developed some systems where people can say "run this tonight" fairly easily. I think that having a terminal at home, I can do some of the heavy stuff at night too.

Pragma: Are you crisis driven, or do you have a very formal approach to projects?

Kowarsky: Some of both. There are definitely long term plans and objectives in effect and that I work on steadily with timeframes and deadlines and the cooperation of a large group of people, and all that sort of thing. And those have been some of the most effective things that we've accomplished. But there's always crisis driven things. Every day there's something. Today the accounts payable detail didn't balance with the general ledger by a material amount, and so that's a crisis. I'm going to have to sit down with the controller and we're going to have to select this and sort this and do this report and do that report and do a whole reconciliation project. There's always something like that.

Pragma: What's your word processing setup?

Kowarsky: There is a word processing system that's a part of the Pick system. It's called Jet. It's screen-oriented word processing. Although I'm not intimately familiar with Wordstar, I think it's in that style. Certainly it's not Runoff style. Runoff is on there too, but we are doing more and more things with Jet. We do our monthly bulletin with it.

Pragma: When a new user starts to use your system, how is their training handled?

Kowarsky: I train everybody to start with. I spend at least three hours with anybody, and in some of the positions I spend much more than that. I might have three or four 2 or 3 hour sessions. I like to cover certain things with everybody, and I want to make sure that it gets covered right.

Pragma: What was your worst data processing disaster?

Kowarsky: Worst? You know you repress those things. Oh, I know what my worst disaster was. My worst disaster was, fortunately, not on my own machine. Fortunately or unfortunately. Oh, God. Oh. This is embarrassing to even talk about. God, horrible to remember. I was supporting one of Gathers' systems in a dial-up situation one time. The former controller and I were doing this together. This goes back more than two years. We were remote. There was a tape on the tape drive and I thought that it was a file save tape and I did a :FILES. We started this :FILES and I managed to wipe out the first several thousand frames of the system. Just totally clobbered it. I mean I was sick, really sick. It was a nightmare of a weekend. We went through these incredible contortions to rebuild that system. We knew all the data was still out there somewhere, so we made a special super-short restore tape that would cover the least amount of data. We scaled down the number of ports it had so hopefully we'd get a tiny SYSPROG on there and everything, and we wouldn't even get into the data. Then we went out there and searched for where all the Master Dictionaries were and found them one by one, and rebuilt the D-pointers. I spent like 14 hours picking up the pieces from what happened by typing in :FILES. We got everything back, but oh, the sickening feeling when you realize what you've done. That feeling of "how could I be so stupid not to check that?" There have been a few things like that.

Pragma: What is one of your successes?

Kowarsky: We have an electronic ordering system, where customers have portable terminals and dial a number and transmit their orders 24 hours a day. We developed an interface where that could be received directly into a port. This is one of those "they said it couldn't be done" things. They said you needed a $10,000 off-line receiver, but we did it. That's been a very, very successful aspect of our data processing. The terminals are the dumbest in the world. It starts and it doesn't stop. It doesn't know if it's talking to a computer or a garbage can. We have our own assembly language mode to receive character by character. We sold that interface twice. That's one of the reasons I find that a clever solution. I'm interested in getting the word out. There's a lot of other things that I feel really good about. Perhaps my best success would be Implementor.

Pragma: Give us the story on your Wizard to Implementor efforts.

Kowarsky: Going back to probably the middle of 1981, ADDS asked me to evaluate development tools, such as WIZARD. I evaluated several. I wrote them a long evaluation, pointing out that I thought WIZARD had the most potential, and also pointing out, in some ways quite specifically, features that I felt needed to be added to it in order to make it a more complete product. They basically agreed with my evaluation and they also agreed with many of the suggestions that I made, so they hired me to implement those suggestions. The suggestions ranged from a number of minor fine-tuning bug-fix type adjustments to the addition of several significant features to the product. For example, WIZARD, as it existed when I evaluated it, did not support concatenated ID structures easily. I thought that any kind of program generation tool for a

Pick oriented system should support concatenated IDs, and that's one thing that I have added. Also, WIZARD did not build in any cross-referencing techniques, and I'm a big advocate of cross-referencing techniques as a very powerful data entry aid. Cross-referencing like the ability to enter a customer number by putting in some part of the customer's name. We've had that feature available on our software for a long time. It was built into our distribution software, it was built into the other software that we had as well. I've always been a big fan of it. I thought that it should be built into any kind of a program generator at a very high level, so that was one of the things that I added. In addition to that, and beyond my original specifications, I expanded the scope of the thing. WIZARD is a program generator. It's a very stand alone thing. You go into WIZARD and you set up this screen and the fields that generate the program and that's it. Implementor is really an effort to be an application development tool beyond simply program generation. It structures the developer through the overall definition of a system, in terms of the files that it's going to contain. It allows you to document those things as you specify them, little bits at a time. Then it includes what is essentially a sophisticated dictionary builder that kind of gives you a running report on the dictionary as you're building it and allows you to document the attributes as you create the dictionary. And then when you go in to create a screen, it draws upon previous definitions that you have made. For example, in creating fields in a screen, you don't have to re-define all the parameters. If you've created this field in a dictionary previously, it'll pull many of the parameters right out of the dictionary.

Pragma: That's what Screenpro does.

Kowarsky: Yes, that's correct. So I've tried to integrate it in that sense. It also does a certain amount of the routine housekeeping that's involved in developing and maintaining a system. And it produces documentation. The stuff I do with Implementor now is much better documented, because Implementor is largely self documenting. It puts it all together: what programs are in the system, what screens are in the system, what files are in the system, what the attribute structure of those files are, in a very clear and well organized way. Plus there were many minor tinkerings to improve it. I cut out unnecessary cursor movements and things like that. Sped it up a little here and there.

Pragma: How much of your time do you estimate went into Implementor?

Kowarsky: I estimate 500 hours. One of the reasons that I got involved in the project is that I'm personally very interested in this whole idea of development tools. I think any programmer should be. I don't like the drudgery part of writing programs. I like the creative intelligence part of writing programs. I feel that from the programmer's point of view, one of the most important benefits of tools like this is that they allow you to apply your intelligence to problem solving rather than to routine drudgery.

Pragma: What's your method of testing your software before you release it, to make sure that it really works?

Kowarsky: It varies with the situation. The system that I'm working with now, I'm pounding the hell out of it. I try to get through every task, if I can, and make sure it does everything that I intended it to do. Almost every time that I've put something into production that I haven't tested, I've regretted it. I've experienced that enough that I'm pretty wary of it now. Occasionally I'll make a minor modificiation to a program and I'll feel very sure of what I'm doing. But even then, those are probably the times that I've been hurt the most. You do something and there's a little doubt. Maybe I should test this? Then you get a little bit lazy and a little bit pressed for time and you slough it off. Almost invariably it comes back to haunt you. That's what I've experienced more times than I would like to admit, so I do try to test everything before I put it into production. If it's a very stand alone thing, it's no big deal. But if it's something that's updating your inventory master file, and you don't test it, you're very likely to be very sorry.

Pragma: What items are on your personal software wish list?

Kowarsky: Item locking in BASIC. That's a biggie. It bothers me a lot. I think the function is necessary, I really do. One of the problems with group locks is there are only so many of them. So you can run out of them. And I get unnecessary collisions. In the Pick implementation, there's no LOCKED clause in the READU. At the very least, they've got to put a LOCKED clause in the READU. I think it's ridiculous to have a multi-user system as sophisticated as this that doesn't have item locks. It really gets me mad. There's another one too. There should be no such thing as a soft group format error. The system should prevent it from happening. At least it should automatically stop and retry before it gives you the message. I don't know if you have much of the problem on Microdata. It seems to me I do see that more on the Mentor. It happens when two people are updating the same group at the same time.

Pragma: What really is "the same time"? A process has started an update and the time slice is used up?

Kowarsky: That's a very good question. I can't answer it, but there should be enough lock self-protection at that level of the operating system that it doesn't happen at all. It confuses the hell out of the user.

Pragma: Because of the message on the screen?

Kowarsky: Well, the Mentor has a GFE handler, which is very nice. It's much nicer than the Microdata. But when you get a group format error, hard or soft, you go into the GFE handler, so now all the users have to know how to manipulate their way out of that. And it's not a friendly process. If the user doesn't do the right thing, the user can hop out of the process in the middle of an update, that kind of thing. You lose some integrity. I haven't had too many really disastrous consequences, but it makes me mad. It shouldn't happen.

Pragma: What do your superiors want to see?

Kowarsky: They want to see information that will facilitate Rainbow making a lot of money. The main priority from their perspective right now is management information with which the entire operation can be fine tuned.

Pragma: What do your assistant and your users want to see out of the data processing function?

Kowarsky: Number one for many of them: response time. Overall, I think what everybody wants to see is things that will make their jobs a little bit easier.

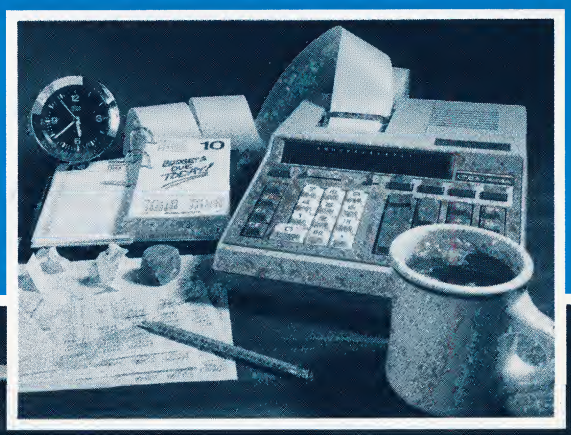
Pragma: How do you think you compare to other sites?

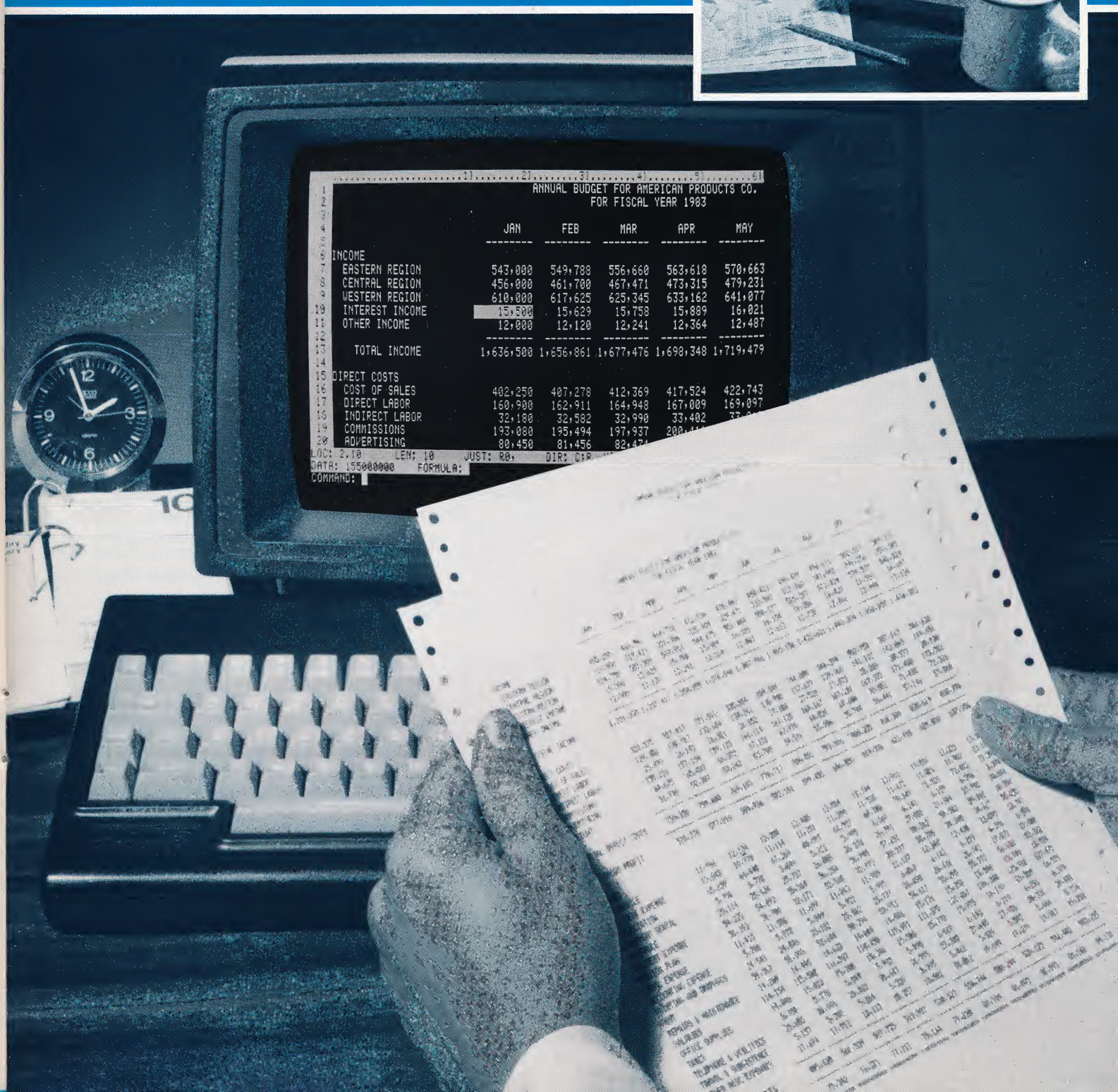
Kowarsky: I think that we are a pretty sophisticated distributor. We're not the most sophisticated, but we're ahead of the average.

Save hours of frustrating manual calculations

COMPU-SHEET

The management tool for anyone who needs a faster, easier way to solve problems.





What is COMPU-SHEET?

Recently, electronic worksheet programs rapidly spread throughout the computer marketplace. Programs such as VISI-CALC®, MULTI-PLAN™ and others have become among the most successful software products ever developed.

There is good reason for the success of these programs. First, they allow the user to simulate everyday business planning problems on their computer in a form that is both easy to use and easy to understand. Second, these tools allow anyone to utilize the computer for complex problem-solving without the technical expertise that was previously required. Now people without programming experience can design, enter and solve in minutes, problems which would take them days or weeks to complete using a pencil and calculator as they have done in the past.

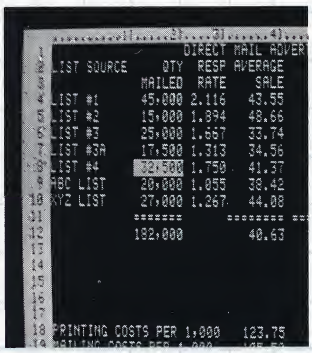
Now, Interactive Systems has developed COMPU-SHEET, an electronic spreadsheet that offers the features of those powerful tools, PLUS the advantage of interfacing with the powerful PICK Operating System. COMPU-SHEET is specifically designed for non-programmers (although experienced programmers will find themselves utilizing COMPU-SHEET too). There is no reason to purchase a separate computer to run your electronic spreadsheet—COMPU-SHEET will run on your existing hardware and can be shared by all users on your system.

COMPU-SHEET gives you, in effect, a large blank sheet of "paper" where you can define and solve your problem. This sheet is divided into columns and rows. You define the number of columns and rows, the width of each column, and the format of the data (such as: whether the information is to be left or right justified, how many decimal points should be displayed, should dollar signs and credit symbols be displayed, etc. . .). At any time you may insert or add new columns or rows, or you may change the width or format of any column.

VISI-CALC® is a registered trademark of VisiCorp. MULTI-PLAN is a trademark of Microsoft Corporation. COMPU-SHEET is a trademark of Interactive Systems. PICK and PICK Systems are trademarks of PICK Systems, Irvine, CA.

At any location (the point where a column and row intersect) you may begin to define your problem. A location may contain one of several types of entries.

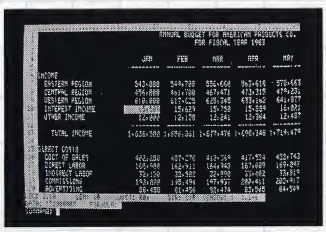
First, it may contain "heading" information. These are entries such as worksheet or column headings ("ANALYSIS OF LEASE VS. PURCHASE" or "SALES," "BUDGET," "VARIANCE," etc. . .). A large heading may spread across several columns.



		DIRECT MAIL MOVER		
LIST SOURCE	QTY MAILED	RESP RATE	AVERAGE SALE	
LIST #1	45,000	2.116	43.55	
LIST #2	15,000	1.894	48.66	
LIST #3	25,000	1.667	33.74	
LIST #3A	17,500	1.313	34.56	
LIST #4	32,500	1.750	41.37	
ABC LIST	20,000	1.055	38.42	
XYZ LIST	27,000	1.267	44.08	
	182,000		40.63	
PRINTING COSTS PER 1,000			123.75	

Second, a location may contain any type of "data." This is virtually any type of information you may choose to enter. It may be a department name, product description, budget amount, sales amount, percentage, or any other type of alphabetic or numeric information you might need. The information entered here may be used in calculations elsewhere on the worksheet.

Third, a location may contain a formula. This formula is written in a simple algebraic format.The formula may operate upon any combination of locations, may contain any type of "constant" values, may retrieve information from any file in your data base (such as your general ledger files, sales files, financial reporting files, etc..), or may retrieve information from other worksheets created by COMPU-SHEET. You may choose to merge information from several worksheets into one "consolidated" worksheet. The power of COMPU-SHEET is limited only by your imagination.

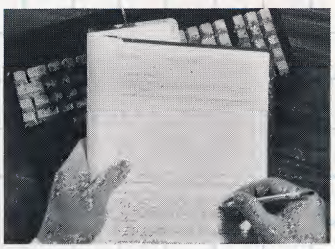


Once your worksheet has been created, you can start analyzing in ways which were never available before. WHAT IF the interest rate is 11.5%?, 13%?, 15%? . . . WHAT IF sales increase by 18%, materials by 14% and payroll by 12% . . . WHAT IF the response rate is 1.5%, 2.0%, 2.5% . . . WHAT IF . . . . . ???

COMPU-SHEET is truly a MANAGER'S tool. It allows the manager to take advantage of the computer in the important functions of PLANNING and CONTROLLING.

Is COMPU-SHEET easy to learn?

Yes. COMPU-SHEET was designed for non-programmers. There is a detailed reference manual which will step you through each available operation. In just a few hours you will be building meaningful worksheets.

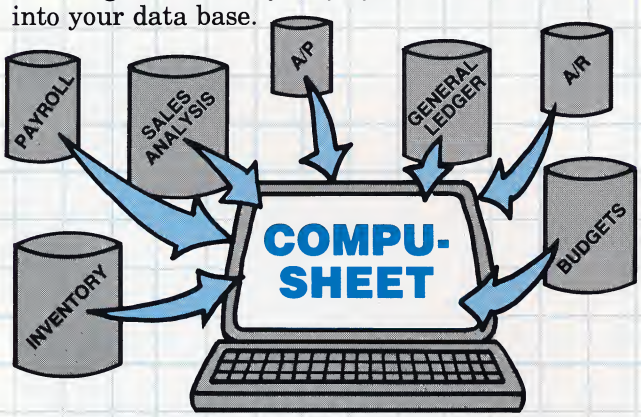


COMPU-SHEET utilizes English-like commands, not cryptic and difficult-to-learn command codes. The computer prompts you each step of the way. But, if you run into trouble, just enter a "?" and COMPU-SHEET will display an explanation of what is needed. COMPU-SHEET is easy to master, easy to use, yet powerful.

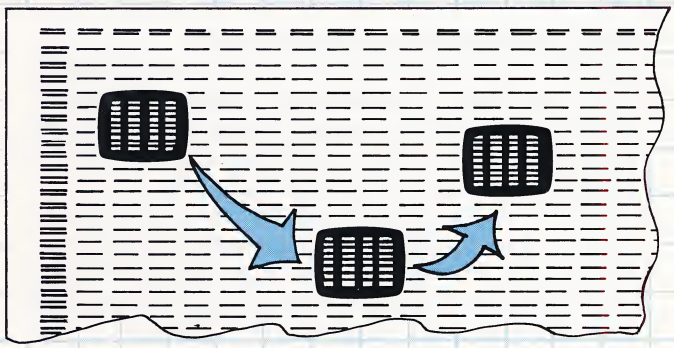
Can COMPU-SHEET handle my problems?

COMPU-SHEET places few limitations on you. There is no limit to the number of columns or rows you may use. Worksheets up to 32,000 characters in size may be created. And if this is still not large enough to handle your needs, you may link them together where information is transferred between any number of worksheets.

Perhaps one of the most significant capabilities of COMPU-SHEET is its ability to retrieve information from your data base. You can link information from anywhere in your system into your worksheets for very advanced analysis and projections. Powerful? Yes, yet it is very easy to do. Imagine the many ways your models can tie into your data base.



One way to perceive COMPU-SHEET is to view your screen as a "window" which displays part of a much larger worksheet. This window may be moved around the worksheet to "uncover" any



section you wish to see. You may divide your screen into several windows, each displaying a different part of the worksheet so you can enter or change data in one section and watch the results in other parts as the sheet is recalculated. Any of these windows may be "locked" in place while you scroll through others.

When you're ready to print your worksheet you will find COMPU-SHEET unmatched in its flexibility. You may ask to print the worksheet and COMPU-SHEET will automatically break it into as many pages as is necessary. You have the option to specify the size of each page. If you want, you can specify any combination of columns and rows to be printed (you can even specify that certain columns and rows are to be repeated on multiple pages).

Formulas may be as simple or as complex as you need to do the job. All standard algebraic operations are available plus a number of expanded operations. Even conditional statements (IF-THEN-ELSE) are supported and gives COMPU-SHEET the capability to handle complex relationships where advanced analysis is required.

Where can I use COMPU-SHEET?

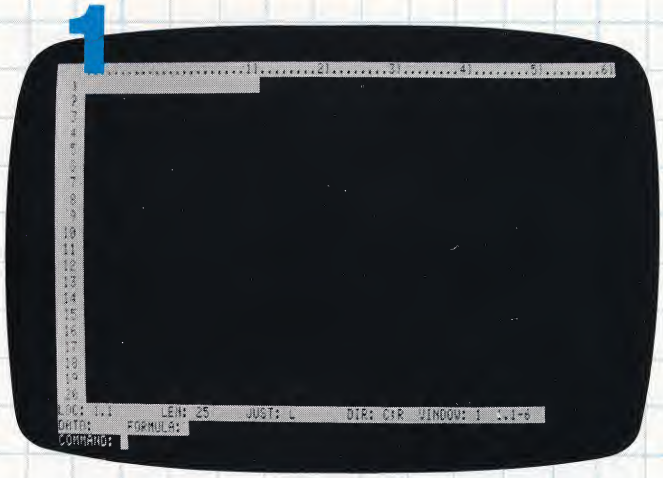
COMPU-SHEET can be used for cash flow analysis and projections . . . Pro-forma statements . . . Budget analysis . . . Profitability analysis . . . Job costing . . . Estimating . . . Manpower assignments . . . Sales forecasting . . . Advertising analysis . . . Vehicle operating costs . . . Trust funds . . . Proposals . . . Labor and material requirements planning . . . the list is endless!

As you can see, the uses of COMPU-SHEET are limited only by your imagination. Many everyday tasks can easily be handled by COMPU-SHEET. Most importantly, think of the analysis you will now be able to do . . . analysis which really NEEDS to be done. With COMPU-SHEET, you now have the time and the means.

How to use COMPU-SHE

After specifying the worksheet size, COMPU-SHEET will display a blank worksheet. In this example, the first six columns (numbered at the top) and twenty rows (numered on the left side) may only be part of a much larger worksheet. What you see here is a "window" on the worksheet. The window may be moved around so you can see any part of the worksheet. At the bottom of the screen are the "status" lines. These lines keep you informed of important information concerning current location size and format, window size, and any previously entered formulas. The last line is the "command" line where all commands are entered.

1

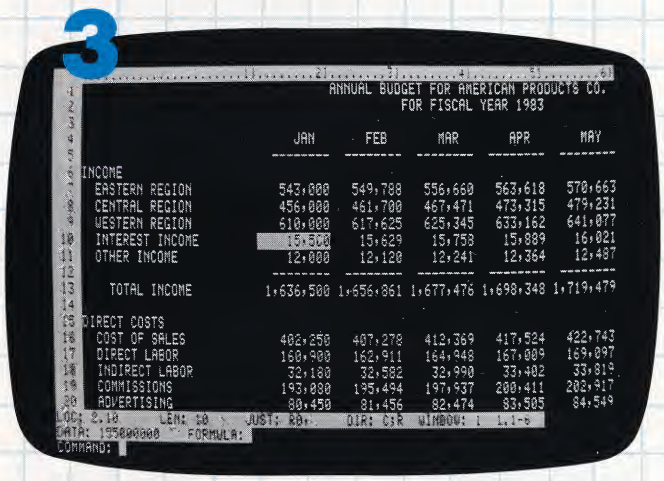


To begin building your worksheet, you simply start entering information. Here, we are entering the worksheet heading and column and row headings. A large heading (such as the worksheet name) may extend across several columns. Since the width of each column may vary, your worksheets may be designed to fit the unique requirements of each application. There is no restriction on the contents of any location. A location may contain a heading, any alphabetic or numeric information, or a formula which specifies any calculations which are to be performed to yield the needed results.

2

Here we have finished entering the data and formulas. COMPU-SHEET automatically calculates each formula as it is entered so you can test the worksheet as you are building it. Formulas may reference a location as either "absolute" (meaning a specific location) or "relative" (meaning a specific number of locations away from the location being calculated). By using this method of entering formulas, you may enter a formula once and then copy it to a range of other locations (this offers significant time savings in entering formulas). Any of these formulas may retrieve information from other worksheets or any file in your system.

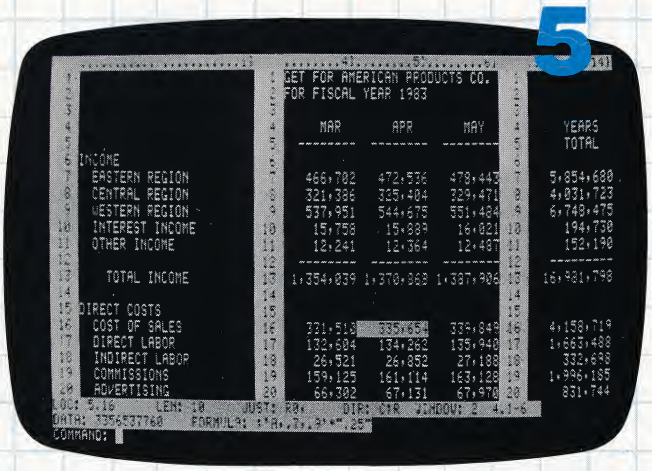
3



HEET

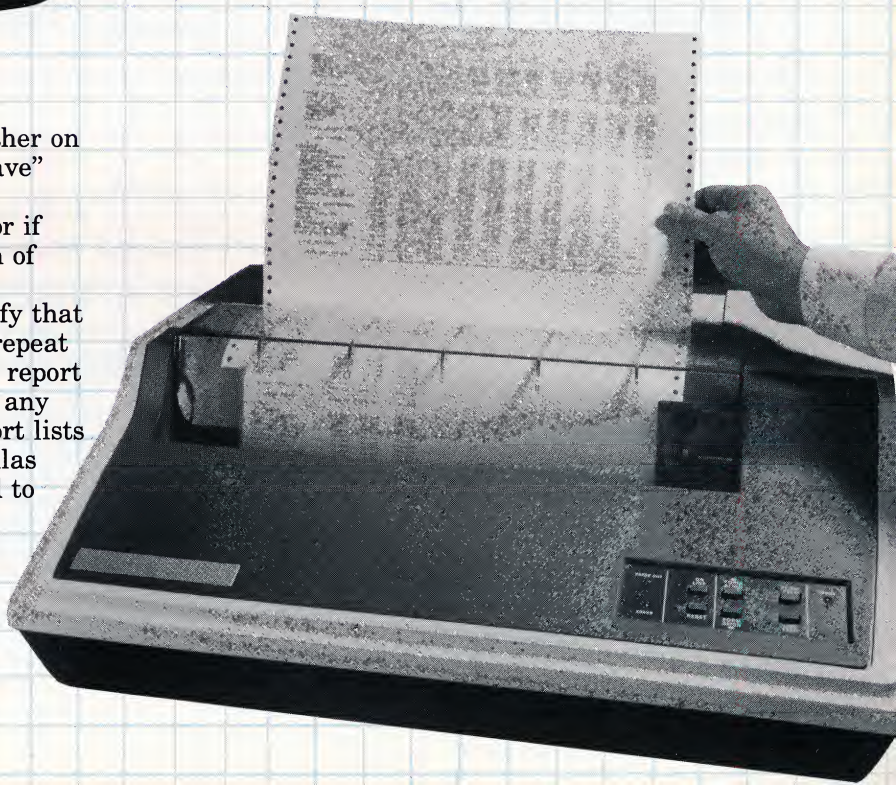
Here is a step-by-step example of a typical COMPU-SHEET analysis. The basic procedure is as follows:

Now that the worksheet is completed, you can begin playing "WHAT IF???". By simply changing any number(s) on your worksheet, you may recalculate another approach to the same problem. This method of analysis saves you many hours of manually testing out a number of approaches to your planning problems. Of course, you are free to change any formula, insert or delete columns or rows (formulas are automatically adjusted if desired), change the name of the worksheet and save a number of variations of the analysis.



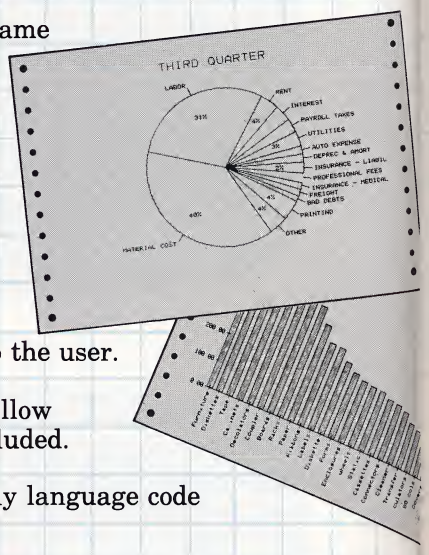
Sometimes, when you have a large worksheet, you may wish to view several sections at the same time. In this example we have defined three "windows" each displaying a different part of the worksheet. You can change one window and watch the results in the others. You may define as many windows as will fit on the screen. Any window may be "locked" to prevent scrolling either vertically or horizontally so you can continue to view it while you scroll through others.

You may choose to print the worksheet either on the line printer or on a terminal driven "slave" printer at any time. Large worksheets will automatically "fold" across multiple pages or if you choose, you may select any combination of columns and rows to print (if the selected worksheet can't fit on a page you may specify that selected heading columns and rows should repeat on multiple pages). There is also an "audit" report available for either the entire worksheet or any combination of columns and rows. This report lists the contents of each location and any formulas entered. It is a valuable tool when you need to audit your worksheet for problems or for documentation justifying your results.



A Summary of COMPU-SHEET features

- Very easy to use . . . no programming experience is needed.
- Uses simple English-like commands (COPY, MERGE, PRINT . . .), not cryptic command codes.
- In-screen prompting for data . . . the cursor moves in any direction over the worksheet.
- Automatically adjusts formulas during insert and delete functions.
- Formulas provide for "absolute" or "relative" reference to worksheet locations.
- Insert or delete multiple columns or rows with one command.
- User may specify titles to appear anywhere on the worksheet.
- Worksheets may be merged or consolidated. The user controls the range of locations to be merged or consolidated.
- Allows for any number of columns or rows up to a worksheet size of 32,000 characters.
- Formulas may specify retrieval of data from other worksheets.
- Formulas may specify retrieval of data from any file in the user's data base (down to the sub-value level, if desired).
- Rapid movement to any point in the worksheet.
- Formulas are validated upon entry. No need to wait until the worksheet is calculated.
- The user may enter a "?" at any point to get a "HELP" display.
- Multiple windows provide for viewing various sections of the worksheet at one time.
- The user controls the width of any column and may change it at any time.
- Any column may be either left or right justified and the masking is set by column (individual locations may override the column mask).
- Worksheets may be password encoded for security.
- The user has the option to control the format of any printed reports. Any combination of columns and rows may be selected.
- Formulas are written in simple algebraic format. Even "IF condition THEN formula1 ELSE formula2" is supported.
- A complete and formatted "AUDIT" report is available.
- All calculations and intermediate results are carried to four decimal place accuracy to insure that rounding errors will not distort the end results (even though displayed values may have less than four decimal places).
- Supports a line printer or terminal driven "slave" printers.
- User coded subroutines may be called from the formulas (for users with technical expertise).
- Multiple versions of the same worksheet may be saved.
- Interfaces with "ACCU-PLOT" graphics system. Your worksheets may be represented as line graphs, pie charts, bar graphs, etc . . .



- Source code is provided to the user.
- A complete and easy-to-follow instruction manual is included.
- Does not use any assembly language code (user modes).
- Operates on Microdata, all PICK systems, and the Prime Information System.
- Operates on any terminal utilizing standard cursor control.
- All revisions and enhancements are provided free of charge for one year.
- A 21 day free trial period is available.

How to order COMPU-SHEET

You can order COMPU-SHEET with a free 21-day trial period. Here's how:

1) Complete the license agreement below.
2) Send the agreement along with your check for $795.00 to Interactive Systems or your local COMPU-SHEET dealer.
3) We will send your copy of COMPU-SHEET (magnetic tape and a complete instruction manual) within 15 days by registered mail.
4) We will hold your check for 21 days from the date you receive COMPU-SHEET.
5) If for any reason you are not completely satisfied with COMPU-SHEET, within that 21 day period call us and we will authorize you to return COMPU-SHEET. We will then return your UNCASHED CHECK to you.

All revisions and enhancements will be provided to you free of charge for a period of one year from the date of purchase.

COMPU-SHEET brings the power of the computer to your imagination and fingertips. Complete and mail the license agreement today!

COMPU-SHEET NON-EXCLUSIVE SOFTWARE LICENSE AGREEMENT

Licensee:

Company _____

Contact Name _____

Address _____

City _____ St _____ Zip _____

Phone _____

Computer Equipment:

Manufacturer _____

Model _____

Serial number _____

Operating system version _____

Tape density _____

INTERACTIVE SYSTEMS will provide the software product COMPU-SHEET on a non-exclusive perpetual license to the Licensee for use only on the Licensee's Computer System described above.The Licensee understands and agrees that COMPU-SHEET constitutes Proprietary Information and Trade Secrets of INTERACTIVE SYSTEMS and that title and ownership of COMPU-SHEET remains with INTERACTIVE SYSTEMS. The Licensee hereby agrees that COMPU-SHEET may not be copied, sold, rented, leased, given, assigned, or otherwise made available to any other individual or organization not licensed by this agreement. INTERACTIVE SYSTEMS warrants COMPU-SHEET to be free from software logic errors for a period of 1 (one) year. INTERACTIVE SYSTEMS makes no warranty, express or implied, concerning the applicability of this Software for a particular purpose. It is solely the Licensee's responsibility to determine suitability of COMPU-SHEET for a particular purpose. INTERACTIVE SYSTEMS accepts no liability for loss or damage caused, or alleged to be caused, directly or indirectly, by COMPU-SHEET. THE WARRANTIES CONTAINED IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY REGARDING MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

License Fee
$795.00 per system

Accepted for Licensee by: _____

Title: _____ Date: ___ / ___ / ___

INTERACTIVE SYSTEMS ■ 129 East Voltaire Avenue ■ Phoenix, Arizona 85022 ■ (602) 993-3579

Other software tools available from INTERACTIVE SYSTEMS:

TCL-AUDITOR is a TCL pre-processor which provides each terminal user with a list of the previous 64 sentences entered at TCL. Any of these previous sentences may easily be listed, corrected, modified, expanded, re-executed or saved without re-entering the sentence.

REPORT-GEN
With REPORT-GEN, you can generate a BASIC program from an ENGLISH sentence, and then modify the program to perform tasks that ENGLISH cannot handle.

SCREEN-GEN is a utility that allows *anyone* to develop data entry or inquiry processes such as order entry, journal entry, customer inquiry, etc. SCREEN-GEN includes an ENGLISH-like language that makes *data entry* as easy as ENGLISH makes data retrieval or report generation.

COMPU-SHEET

A new electronic worksheet from Interactive Systems

For Microdata, all PICK systems, and the Prime Information System.

Selected by CDI for each IBM Series/1.

Selected by General Automation for each ZEBRA (PICK) system.

Available from Interactive Systems and dealers around the world.

For information, contact:

INTERACTIVE SYSTEMS
129 East Voltaire Avenue
Phoenix, Arizona 85022

COMPU-SHEET

A management tool for company presidents, CPA's, treasurers, controllers, managers, or anyone who needs an easier way to solve problems!

benchmarks

Are you thinking of doing an upgrade to your hardware or software? Are you comparing throughput and performance while shopping around for a system? Have you converted from one machine to another? Be sure to send in your benchmark statistics to Pragma, so the results can be featured in this department and shared with other installations.

More BASIC Comparisons

More data from other machines running LOOP vs. LOCATE, plus a comparison of the OCONV and FIELD functions.

Last issue's benchmarks generated a lot of interest among Pick users, both inside and outside the Microdata community. A consequence of some of the calls and letters about those timings are two new sets of numbers, a result of running the same benchmark on two non-Microdata machines: the IBM Personal Computer and the ADDS Mentor. During our interview for this issue's User Profile, the LOOP and LOCATE programs were discussed, and Steve Kowarsky agreed to duplicate the tests on the Mentor at Rainbow Natural Foods, resulting in a time of 114 seconds for LOOP and 44 seconds for LOCATE. Roger Harpel of Cosmos reported times of 133 seconds for LOOP and 32 seconds for LOCATE, which were obtained by using his company's single user Pick compatible Revelation operating system for the Personal Computer.

The listings below are new tests in the theme of comparing two different methods of doing the same operation with a BASIC program. In order to compare the difference in speed between the FIELD and OCONV functions when extracting substrings, the two programs shown were executed on a Microdata 6000 with MOS memory. The results are that the FIELD function is substantially faster than OCONV. The FIELD version finished in 39 seconds, while the OCONV version executed for 62 seconds. Both functions are often used for substring extraction by BASIC programs. For some programmers, the order and especially the format of the OCONV arguments are perhaps easier to remember, since they duplicate the conversions so frequently used in ENGLISH. But FIELD, besides executing faster (if that should make a difference to an application), also allows use of calls to the COL1() and COL2() functions. The difference in speed may be because FIELD is designed for just one type of operation, while OCONV is capable of doing any conversion, but that is just a conjecture.

```
    FIELD.BENCH
001 START = TIME()
002 FOR I = 1 TO 10000
003   MIDDLE = FIELD("LEFT*MIDDLE*RIGHT","*",2)
004 NEXT I
005 PRINT (TIME()-START):" SECONDS"
006 STOP
007 END
```

```
    OCONV.BENCH
001 START = TIME()
002 FOR I = 1 TO 10000
003   MIDDLE = OCONV("LEFT*MIDDLE*RIGHT","G1*1")
004 NEXT I
005 PRINT (TIME()-START):" SECONDS"
006 STOP
007 END
```

```
MRB.STOCK : 124644
REF   8669
PART#   000083001
QTY ISSUED   2
TRANS-CODE   SK
DATE   09-01-82
UNIT-COST   54
WORKORDER NUMBER   8669
TYPE   P

MRB.STOCK : 129441
REF   9167
PART#   480001320
QTY ISSUED   8
TRANS-CODE   SK
DATE   11-16-82
UNIT-COST   0
WORKORDER NUMBER   9167
TYPE   P

MRB.STOCK : 134238
REF   29681
PART#   470000034
QTY ISSUED   1000
TRANS-CODE   PO
ACCT#*DEPT   15
BATCH-PO#   4117*5
DATE   01-13-83
UNIT-COST   0.0100

MRB.STOCK : 122265
PART#   802041000
QTY ISSUED   3
TRANS-CODE   EN
DATE   07-20-82
UNIT-COST   48

MRB.STOCK : 127062
REF   28102
PART#   480604640
QTY ISSUED   1120
TRANS-CODE   DS
BATCH-PO#   3639
DATE   10-14-82
UNIT-COST   0

MRB.STOCK : 131859
PART#   000082000
QTY ISSUED   12
TRANS-CODE   SK
DATE   12-08-82
UNIT-COST   1
WORKORDER NUMBER   9303
TYPE   P

MRB.STOCK : 124645
REF   1312
```

```
MRB.STOCK : 124644
REF.................   8669
PART#...............   000083001
QTY ISSUED..........   2
TRANS-CODE..........   SK
DATE................   09-01-82
UNIT-COST...........   54
WORKORDER NUMBER....   8669
TYPE................   P

MRB.STOCK : 129441
REF.................   9167
PART#...............   480001320
QTY ISSUED..........   8
TRANS-CODE..........   SK
DATE................   11-16-82
UNIT-COST...........   0
WORKORDER NUMBER....   9167
TYPE................   P

MRB.STOCK : 134238
REF.................   29681
PART#...............   470000034
QTY ISSUED..........   1000
TRANS-CODE..........   PO
ACCT#*DEPT..........   15
BATCH-PO#...........   4117*5
DATE................   01-13-83
UNIT-COST...........   0.0100

MRB.STOCK : 122265
PART#...............   802041000
QTY ISSUED..........   3
TRANS-CODE..........   EN
DATE................   07-20-82
UNIT-COST...........   48

MRB.STOCK : 127062
REF.................   28102
PART#...............   480604640
QTY ISSUED..........   1120
TRANS-CODE..........   DS
BATCH-PO#...........   3639
DATE................   10-14-82
UNIT-COST...........   0

MRB.STOCK : 131859
PART#...............   000082000
QTY ISSUED..........   12
TRANS-CODE..........   SK
DATE................   12-08-82
UNIT-COST...........   1
WORKORDER NUMBER....   9303
TYPE................   P

MRB.STOCK : 124645
REF.................   1312
```

Justifying Ragged Output

A program is presented for padding attribute headings to a standard length.

The column of output on the far left is a typical sample of the results obtained when given a LIST or SORT command with no output specifications, and the dictionary for the file has a good number of default attribute definition items with identifiers that are numeric and sequential (1, 2, 3...). There tend to be too many columns for the width of the output device, so the system automatically outputs the listing with the column headings in vertical, non-columnar form. Unfortunately, the resulting output is very ragged and difficult to read.

The second sample column shows the exact same data, but with all attribute labels padded to the same length with periods, giving a justified column that is much easier to read. This is accomplished by executing the following program:

```
    MAKE.DOTS
001 PRINT "JUSTIFY DICT OF WHAT FILE":
002 INPUT FILE
003 OPEN "DICT", FILE ELSE STOP "CAN'T OPEN!"
004 ID = 1
005 100 READV HEADER FROM ID, 3 ELSE STOP
006 HEADER = HEADER : STR(".",20-LEN(HEADER))
007 WRITEV HEADER ON ID, 3
008 ID = ID+1
009 GO TO 100
010 END
```

The MAKE.DOTS program accepts the name of a file and pads the heading string in attribute 3 of each default attribute definition item in the dictionary of the file. The program pads dictionary words 1, 2, 3 and so on until a numeric item identifier is not found in the dictionary. Line 6 of the program controls both the character used for padding and the resulting width of each label (up to 20 periods in this version).

A variation of MAKE.DOTS that some users find preferable has the concatenation operands in line 6 reversed, so that the label text is right justified.

P

wish list

Users of computer systems should always remember that the nice thing about software (besides the fact that it doesn't break when you drop it) is that programs are relatively easy to change — no soldering gun is necessary. So just because the operating system or the compiler or a system utility happens to work (or fail to work) in some particular fashion now, doesn't mean it has to always be that way. The next time an inspired idea arrives for improving your system, write it down and send it to Pragma for publication in the Wish List. Naturally, all such submittals are eligible under Pragma's author payment program.

The previous 48 Wish List items have been featured in issues #1 and #2 of Pragma.

49. S/NAME Attribute Headings. S/NAME strings may only be literal constants, so that headings like "QTY THIS MONTH" must be used instead of "APRIL QTY" (which would have to be edited each month). Allow computable expressions (such as A-correlatives) to be used as ENGLISH column headings.

50. Editor DE Command. If the user wants to delete lines 537 through 824 with the Editor, the user must first go to line 537, compute the total number of lines being deleted (which means subtracting the two line numbers and adding 1), and then give the command DE288. Instead, allow a range of line numbers to be specified in the DE command, so that the user only needs to input something like DE537-824.

51. Item Identifiers of Any Length. Usually, the limit of 50 characters for item identifiers presents no problem, but there are times when it would be convenient to be able to break the rule. Allow identifiers to be any length.

52. Finding Owners of Group Locks. Even when a READU statement uses the LOCKED clause to report that a group is locked, the operator must go to the TCL and use the ITEM and LIST-LOCKS verbs to determine who (which port) has the group locked. Provide a way for programs to determine which port has a particular group locked.

53. Attribute Security. Support the use of security codes in the L/RET and L/UPD attributes of dictionary words, so that attribute definition items can be constructed that will allow access and updates for certain attributes only from certain accounts.

54. ENGLISH Headings that Document. If an ENGLISH command does not include a HEADING clause, then a default heading that shows only the page number, date and time is generated for the report. It would be much more useful for the default heading to also include the text of the command itself that generated the report. In this way, ENGLISH reports become self-documenting. A glance at the header on any report page will tell the reader what command can be used to duplicate the report.

55. Spooler Accounting. The system keeps careful records of an account's use of CPU time, disk reads, and real time, but there is no way to know how much paper and printing time an account consumes. For each account, have the spooler keep track of the number of lines printed, pages used, and forms changed. The latter would be recorded to help charge for operator labor and for the different costs involved for different forms.

56. Group Lock Limit. The system can not lock more than a given number of groups at a time. Unfortunately, a single program often needs to lock a number of files at once. A system with a large number of terminals executing a variety of programs may reach the limit on the number of locks allowed. Allow any number of groups to be locked simultaneously.

57. Detecting the Progress of Slow Commands. Once a time consuming ENGLISH command has been given, the operator can only sit and wait for control to return. It is frustrating not knowing how far along the command is or how long it will take to complete. Give ENGLISH the option of reporting its progress (perhaps in terms of items checked, and/or items selected, and/or lines output) while processing commands.

58. Support MF Conversions Everywhere. The MF conversion code is a powerful tool, but only allowing it to be invoked from ENGLISH is a serious limitation. Allow MF conversions to be used by DATA/BASIC, Screenpro, and procs.

59. Select Just One. There are occasions when the user is interested in finding only the first occurrence of any item that meets a given selection criteria. Allow a modifier in ENGLISH, for example the word "FIRST", which when included in a command indicates a request for a select list of just the first item found that meets any specified selection criteria, so that the system does not bother scanning the rest of the file.

60. Time Limits on User Input. It is often desirable to allow the operator only a certain amount of time to respond to a prompt for input. Games, simulations, diagnostics, unattended batch runs and similar programs need the ability to regain control if a prompt goes unanswered for a certain amount of time. For every type of statement in the system that accepts input, allow a version that regains control (perhaps through a type of ELSE clause) if the prompt goes unanswered, where the amount of time to wait can be a variable.

61. Abolish Editor F Command. The F command is more of a convenience for the Editor than it is for the user. Design the Editor so that text changes are always immediately updated, the "SEQN?" message is never displayed, lines can be edited in any order, and the F command is obsolete.

P

command files

Command files are typically the "glue" that holds the components of a software system together. Microdata's command file capability is called PROC, Prime offers Perform, and so on. In this regular department, Pragma will be presenting concepts and techniques to help installations squeeze the most out of their command file processors.

Generating Monthly Column Headings

A proc is presented, demonstrating the ability to generate reports featuring current month names for column headings, without relying on BASIC programming.

The sample January report below is a typical requirement of many data processing installations: the need for a report where the vertical columns are "rolling" time periods that rotate left each time the report is generated in the next reporting period. For example, this report would need to drop the January (1983) column from the leading left side and include a January (1984) column on the trailing right side when it is output again in February (1983).

With a report generator like ENGLISH, column headings are defined as string constants, so it is relatively easy to set up column headings for the appropriate month names for any given run. But to have the month names automatically rotate each month, without programmer intervention, is a more difficult problem. Sometimes an installation will avoid the problem by forever labeling the columns "CURRENT MONTH", "NEXT MONTH", "MONTH 3", and so on, but this is not very user friendly.

The proc on page 29 shows a solution to the problem. The proc uses dictionary words that have no column headings defined, so that the columns are simply headed with a string of periods which serve as underlining for the "real" column headings, which are constructed by the proc as part of the HEADING clause. Line 23 of the proc is a 24 month list of month names, from which the proc can extract a string starting with any current month, and the string is guaranteed to include the following 11 month names. Each month the proc is executed, it automatically constructs the appropriate column headings.

The file used in this particular example is a parts file, where every item identifier is a part number. The file happens to be constructed such that attribute 14 is a multivalued list of receipt dates (coded in internal form as integers), and attribute 15 is an associated list of receipt quantities for those dates. There is no special reason why the data must be stored in those particular attributes, so that the techniques demonstrated here can easily apply to any file with multivalued dates and associated data lists. In this example, the dates happen to be expected part receipt dates, but they could just as easily be payment due dates or some other kind of calendar data, just as the associated quantities could instead be dollar amounts. In any case, page 29 also lists the various dictionary words (in I-DUMP format) used by the proc. Each dictionary word that generates a quantity column automatically selects only those quantities for the month the column happens to currently represent. Note that the correlatives are coded so that the quantity for any date in the current month, even those prior to the day the report is generated, will appear in the first named month column. In other words, the "Past" column means "prior months", not "any date before the report date". P

```
Page 1 --- Scheduled Receipts for Parts by Month --- 03:05:39  24 JAN 1983

Part Number  Past   Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec    Future Total
.....------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------

11111             10     20     10                                                                            40
22222                    15     15                                                                            30
33333                                                                  5     10           15                   30
44444                                                                              10     10     10     10     40
55555                                                                                                  300    300
```

```
001 PQN
002 HSORT PO.PARTS
003 H UNTITLED.PART
004 H PAST.PO.QTY
005 H MONTH0.PO.QTY
006 H MONTH1.PO.QTY
007 H MONTH2.PO.QTY
008 H MONTH3.PO.QTY
009 H MONTH4.PO.QTY
010 H MONTH5.PO.QTY
011 H MONTH6.PO.QTY
012 H MONTH7.PO.QTY
013 H MONTH8.PO.QTY
014 H MONTH9.PO.QTY
015 H MONTH10.PO.QTY
016 H MONTH11.PO.QTY
017 H FUTURE.PO.QTY
018 H TOTAL.PO.QTY
019 H ID-SUPP
020 H HEADING "Page
    'P' --- Schedule
    d Receipts for P
    arts by Month --
    - 'TLL'
021 HPart Number
022 HPast
023 IBH%1:F;C Jan
     Feb     Mar     A
    pr     May     Jun
        Jul     Aug
      Sep     Oct
    Nov     Dec     Ja
    n     Feb     Mar
       Apr     May
     Jun     Jul     A
    ug     Sep     Oct
        Nov;C1;D(D2-
    ]G-1);-;C7;*;C84
    ;[]:
024 A
025 HFuture Total"
026 P
```

```
FUTURE.PO.QTY^S^0^^^^^MDOZ^F;D(D2-]G-1);14(D2-]G-1);];D(D2-]G2-1)
;14(D2-]G2-1);>;*;D(D2-]G2-1);C1;+;14(D2-]G2-1);>;+;C0;#;15;*;S^R
^6^
MONTH0.PO.QTY^S^0^^^^^MDOZ^F;D(D2-]G-1);14(D2-]G-1);=;D(D2-]G2-1)
;14(D2-]G2-1);=;*;15;*;S^R^6^
MONTH1.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C1;+;>;C12;*;D(D2-]G-
1);C1;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C1;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH10.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C10;+;>;C12;*;D(D2-]
G-1);C10;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C10;+;>;D(D2-]G2-1);+;1
4(D2-]G2-1);=;*;15;*;S^R^6^
MONTH11.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C11;+;>;C12;*;D(D2-]
G-1);C11;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C11;+;>;D(D2-]G2-1);+;1
4(D2-]G2-1);=;*;15;*;S^R^6^
MONTH2.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C2;+;>;C12;*;D(D2-]G-
1);C2;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C2;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH3.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C3;+;>;C12;*;D(D2-]G-
1);C3;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C3;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH4.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C4;+;>;C12;*;D(D2-]G-
1);C4;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C4;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH5.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C5;+;>;C12;*;D(D2-]G-
1);C5;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C5;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH6.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C6;+;>;C12;*;D(D2-]G-
1);C6;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C6;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH7.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C7;+;>;C12;*;D(D2-]G-
1);C7;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C7;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH8.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C8;+;>;C12;*;D(D2-]G-
1);C8;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C8;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
MONTH9.PO.QTY^S^0^^^^^MDOZ^F;C12;D(D2-]G-1);C9;+;>;C12;*;D(D2-]G-
1);C9;+;-;14(D2-]G-1);=;C12;D(D2-]G-1);C9;+;>;D(D2-]G2-1);+;14(D2
-]G2-1);=;*;15;*;S^R^6^
PAST.PO.QTY^S^0^^^^^MDOZ^F;C1;D(D2-]G1-1);-;D;-;14;<;15;*;S^R^6^
TOTAL.PO.QTY^S^0^^^^^MDOZ^F;15;S^R^6^
UNTITLED.PART^S^0^^^^^^^L^12^
```

VANILLA The No-Frills Manufacturing System

Part 3: Purchasing

The third in a series of articles on the design and implementation of Vanilla, a software system for manufacturing, is presented. Concepts and requirements for the creation and maintenance of a purchase order file are described.

This is the third in a series of articles devoted to the design and implementation of Vanilla, a software system for a hypothetical manufacturing company. The previous installment in the series (Pragma #2, page 27) presented a program for creating and changing entries in a bill of material file. In this installment, the requirements for updating purchase orders will be the subject for discussion.

A purchase order is the document by which a manufacturer orders and receives purchased parts for use at the lowest levels of assemblies. Maintaining a file of all purchase orders is an important aspect of any manufacturing system that includes MRP, since purchase orders indicate all scheduled receipt dates and quantities for purchased parts — information with which the MRP explosion and netting process can determine the future availability of parts for meeting requirements against inventory.

A company's Purchasing department will typically have a large number of requirements concerning the acceptable content and structure of the purchase order file. Among the items that must be considered when implementing a data processing system that supports a purchase order file are:

VENDORS. *In Vanilla, every purchase order must have a vendor. Every vendor is identified by a unique number, and is named in the vendor master file.*

LINE ITEMS. *In Vanilla, there may be any number of line items per purchase order. Each line item identifies the part number (optional), description, unit of measure, charge account number, order quantity, due date and price for the item being ordered. The same part number may appear in more than one line item. Each line item may have any number of scheduled deliveries. There is a quantity, due date and price for each scheduled delivery. The order of line items in a purchase order is as entered manually by the operator.*

OTHER ORDER ATTRIBUTES. These are typically things like addresses, shipping methods, terms, buyer codes and other bits of information of interest to Purchasing personnel. *Vanilla supports a variety of additional order attributes to facilitate general purchase order maintenance.*

Technically, an MRP system only requires knowledge of purchase order part numbers, due dates and quantities in order to successfully calculate inventory requirements and excesses. However, such a simplified form of purchase order entry would be more of a convenience for the computer than for the Purchasing department, so Vanilla will support a somewhat more comprehensive purchase order file than is strictly necessary for just computing MRP.

In the next installment, the actual program for maintaining the purchase order file will be presented.

queries

If you have questions or if you have answers, send either to Pragma for publication — both are eligible for Pragma's author payment awards.

The previous 18 Queries have been featured in issues #1 and #2 of Pragma.

19. How can one change the descriptions of terminal locations that are displayed by the LISTU command?

The descriptions for terminal locations are kept in the first attribute of certain items stored in the dictionary of the ACC (accounting) file. To edit the descriptions, give a command like "EDIT DICT ACC 00 01 02 03", which would invoke the Editor and let you change the descriptions for ports zero through three. Refer to the Editor manual for information regarding how to actually change text data.

20. If a proc invokes three DATA/BASIC programs in a row, how can the middle program conditionally stop proc execution so that the third program is never invoked?

One method is to let the program execute a CHAIN to a proc which then exits with the X command. Another method is to have the program PROCWRITE some type of flag, and then let the calling proc test the flag in the proc buffer to determine whether or not to invoke the next program.

21. Inspecting the 4.1 DATA/BASIC compiler's own code reveals that statements named PROGRAM and GARBAGECOLLECT are apparently available. Sure enough, the compiler accepts these statements when they appear in a program (the PROGRAM statement requires a parameter), but do they actually cause anything to happen?

22. We have 8-way boards with DIP switches for setting baud rates, as mentioned in Wish List item #18 (Pragma #1, page 28). What are the actual switch settings for the various baud rates?

Setting the 10th switch sets the rate to 9600 baud. Switches 9, 8, 7, 6 and 5 set the rate to 4800, 2400, 1200, 300 and 110 baud, respectively. Switch 1 sets the "stop" bit.

23. What is user exit U508E for?

If U508E is used as a correlative in a dictionary word, then a LIST or SORT using the word will cause every attribute in each item to be displayed in COPY format, while allowing other ENGLISH features, such as HEADINGs, to be used. Brief documentation on U508E originally appeared as part of a 2.3 update package. The U508E exit does not work properly when specified with other conversion codes, or when substring searching with the usual bracket characters is attempted in the ENGLISH command. [P]

An Introduction to ENGLISH

Part 2: More Commands

The second in a series of articles is presented for the beginning user of Microdata's inquiry and report generation language called ENGLISH. New commands for exploring files are given.

This is the second in a series of articles that will be devoted to explaining ENGLISH, Microdata's computer language for retrieving and listing computer files. The last installment *(Part 1: Jargon,* Pragma #2, page 24) explained jargon such as "files", "logging on", and "to input a command". We discovered that **files** are made of **items** that are named by an **identifier,** and that items are made of **attributes.** Did you try the suggested command LIST ACC? Can you guess why the file is named ACC?

Now try the command SORT ACC and then try COUNT ACC. What's the difference between the results that LIST, SORT and COUNT give? The words LIST, SORT and COUNT are known as ENGLISH **verbs.** The first word of any ENGLISH command must always be a verb. What happens if your command doesn't start with a verb? To find out, try the command LEST ACC.

Another word that can be used in commands is ONLY. It fits between the verb and the file name, such as in LIST ONLY ACC. What happens when ONLY is included in a command?

Now for another piece of jargon: every file in the computer has a **dictionary,** which is a collection of words that can be used to control what file attributes get listed. To see the dictionary for the ACC file, use SORT ONLY DICT ACC. Pick a word out of the dictionary for ACC and try it in a LIST command. For example, LIST ACC A1. Try more dictionary words in LIST and SORT commands, maybe even two or three words at a time. What happens? [P]

the computer room

A well-run computer room is the sign of an efficient data processing organization, and an important requirement in any computer room is good documentation — documentation regarding the care and use of peripherals, system error recovery, maintenance procedures, and a myriad of other details regarding day-to-day computer operations. In this regular department, Pragma will be presenting documentation, ideas, and techniques for guaranteeing a smooth-running computer room.

Shared Site Checklist

A checklist is given for installations that agree to share facilities in case of emergency, for purposes of disaster recovery.

Regardless of any planning that is done to help prevent disasters in the computer room, every installation should also have a plan for how to recover from a disaster if it should happen anyway. One way in which an installation can help prepare for disaster recovery is to have a mutual agreement with another installation (with similar equipment) to share computers in cases of emergency. That is, each site promises it will make its computer available to the other, if one should experience some sort of disastrous problem. Here then is a checklist of ideas that should be addressed by any agreement to share computers in cases of emergency.

☐ **Purpose.** Does your agreement state its goals and purpose?

☐ **Definition.** What constitutes an emergency? Does it have to be a fire, earthquake or flood? Or just a disk crash or broken terminal? Is it an emergency when your customer engineer fails to repair a blown tape drive after four days? Can software bugs cause emergencies?

☐ **Warning.** How much warning do you require before you'll turn your system over to someone who needs it for an emergency? Is the agreement fair to both parties? (Just because someone demands 24 hours notice before you want to borrow a system, that is not a promise that they will make it available within 24 hours.)

☐ **Availability.** How often and for how long will the backup system be available?

☐ **Consumables.** Who supplies the paper? Who pays for all the phone calls?

☐ **Use.** Is it all right if the borrower of your computer shows up at the door with racks of tapes and cartons of forms, intending to transfer their whole business operation onto your machine?

☐ **Equipment.** Your machine is down and all your software will only run with your custom terminal featuring color answerback, and all your reports are set up for your serial printer that prints from the bottom up. Will your backup site let you bring your own terminals and printers?

☐ **Personnel.** Can you handle a visiting staff of 14 people using your system while they wait for theirs to be repaired?

☐ **Loading.** When someone needs to borrow your system, do they just mount their file save tape and do a complete restore? Who is responsible for saving and restoring the contents of your disk?

☐ **Reliability.** Should each site be required to inform the other whenever a hardware failure of any type and severity occurs? Or do you care if your backup site has a lemon?

☐ **Configuration.** Should each site be required to inform the other whenever the hardware configuration changes? What if you suddenly need to use your backup, only to discover they converted to an incompatible tape density six months ago?

☐ **Confidentiality.** Is your agreement a secret?

☐ **Term.** How long does your agreement last?

☐ **Responsibility.** Is everything free of charge? What if your backup clobbers your only file save tape? What if both systems go down? What if someone fails to hold up their side of the deal?

Understandings are often based on verbal agreements, but it is probably much better to have promises in writing, because if such an agreement is actually exercised, there can be all kinds of circumstances and conditions which might lead to misunderstanding and confusion, unless all the details have been previously written down and studied. And remember that regardless of the form of an agreement, contracts (whether oral or written) are best approved by lawyers.

P

Converting Paint to Programs

A program to automatically create other programs for printing text created with Screenpro's painter, as though the painter were a general purpose text editor, is presented.

Many Microdata users who have avoided Screenpro do not realize that Screenpro's screen painter is actually a powerful, general purpose utility that can be used in a variety of interesting ways. The screen painter is, in fact, a type of full-screen text editor.

Using the PAINT-SCREEN command is a convenient way to try the painter without getting confused by or enmeshed in other Screenpro functions. For example, the command PAINT-SCREEN MD FORM.TEST will enter the painter and allow the user to paint a screen that will be saved (if the user subsequently gives the painter's F command) in the Master Dictionary under the name FORM.TEST. (Once the painter has been invoked, typing a "?" will display the various available painter commands, and Section 3.7 of the Screenpro manual provides more details on using the painter.)

One useful application for the painter is the creation of labels for a printed form, such as labels for a purchase order or on a payroll check. Programmers will sometimes measure by hand the position of all labels on such forms, and then manually create a BASIC program full of PRINT statements and string constants to duplicate the form on a line printer. An easier approach is to use the painter to interactively create the form on a high-speed terminal's display, then use a program similar to the CODE-PAINT program listed below to automatically generate a BASIC program with all the necessary PRINT statements. In other words, once the PAINT-SCREEN MD FORM.TEST command has been used to create a screen of labels, executing the CODE-PAINT program (and answering the DATA FILE? prompt with "MD", and the ITEM? prompt with "FORM.TEST") will create a new item named BASIC.FORM.TEST, which will be a BASIC program capable of printing the form created with the painter. Note that CODE-PAINT assumes the painted screen contains no double quote (") characters.

With post-processors such as CODE-PAINT, Screenpro's painter becomes a valuable utility for doing much more than just creating Screenpro prompts. Ⓟ

CODE-PAINT PROGRAM GENERATOR

```
001 PRINT 'DATA FILE':
002 INPUT FILE.NAME
003 OPEN FILE.NAME ELSE STOP 'NO SUCH FILE!'
004 PRINT 'ITEM':
005 INPUT ID
006 READV PAINT FROM ID, 5 ELSE STOP 'NO PAINT!'
007 SUB.VALUES = COUNT(PAINT,CHAR(252))+1
008 CODE = ''
009 FOR LINE = 1 TO SUB.VALUES
010   TEXT = PAINT<1,1,LINE>
011   IF TEXT # '' THEN
012     LOOP
013       COL = 1
014       LOOP UNTIL TEXT[COL,1] # ' ' DO COL = COL+1 REPEAT
015       IF COL > 1 THEN CODE<-1> = 'PRINT STR(" ",':COL-1:'):'
016       NON.BLANK.POS = COL
017       LOOP
018         COL = COL+1
019         DELIMITER = TEXT[COL,2]
020       UNTIL (DELIMITER = '  ') OR (DELIMITER = '') DO REPEAT
021       CODE<-1> = 'PRINT "':TEXT[NON.BLANK.POS,COL-NON.BLANK.POS]:'":'
022     WHILE DELIMITER # '' DO
023       TEXT = TEXT[COL,LEN(TEXT)-COL+1]
024     REPEAT
025     END
026   CODE<-1> = 'PRINT'
027 NEXT LINE
028 CODE<-1> = 'STOP'
029 CODE<-1> = 'END'
030 WRITE CODE ON 'BASIC.':ID
031 STOP
032 END
```

letters

If you use a Pick system, Pragma wants to hear from you. Have you developed applications of interest to other users? Do you plan to acquire new hardware? What features would you like to see in Pragma? Are you active in any type of user group organization? Write Pragma today. All letters to the Editors are welcome, and as many as possible will be published in the Letters Department in every issue.

Some Wishes are Answered

Thank you for the complimentary copy of your first issue. I'm sure that all Pick system users appreciate any contact with individuals and publications which talk our Pick language. I liked the Wish List feature especially. I would submit some actual programs, but we are a totally RPL shop and my proc or DATA/BASIC is quite rusty.

A solution for Wish #1, waking a SLEEPing process: Write a unique key into the Master Dictionary just before going to sleep. Set your sleep cycle relatively short. After each sleep cycle, read the Master Dictionary for the existence of the key. If it exists, go back to sleep. If it doesn't, do the next task. The second process would simply delete the Master Dictionary key when it wishes the first process to begin the next task.

A solution for Wish #5, Editor prestore command: To simply repeat a command many times in the Editor, separate each command with an ESCAPE character. If the Wish meant editing the same attribute many times, you have the problem of incorporating the F command between each change and getting back to the attribute. I usually store an F and G command in the P1 prestore so that after each edit I save at least two keystrokes and the necessity of having to remember a line number.

A solution for Wish #8, splitting lines with the Editor: This is not a solution for the faint at heart, but I use it now and then when I don't feel like retyping an extremely long or complex sentence. For example:

```
.I
001 THIS IS LINE 1
002 SPLIT HERE TO MAKE THIS LINE 3
003
TOP
.G1
001 THIS IS LINE 1
.
I 6.86
!CR4 ^SPL=  IT H=  ERE =^ERE^
!G
F
TOP
.L99
001 THIS IS LINE 1
002 SPLIT HERE
003 TO MAKE THIS LINE 3
```

[Always go to the line just before the one to be edited, do a BREAK, type "CR4" and hit LINE FEED. The line being edited will be displayed in four-character segments followed by an equal sign. Every subsequent LINE FEED will display the next four-characters in the line. Stop when you reach the four characters that include the point where you want to split the line. Type a single quote and four replacement characters, but type an attribute mark (displayed as an up arrow) in place of one of the characters to cause a new line at that point. Then hit RETURN, type the G command, and again hit RETURN to leave the debugger and resume editing.] You must be logged on with a user priority of 2. While this looks difficult at first, it is actually quite simple. You are replacing a space or some other throwaway character with an attribute mark. One other use for this technique is merging two lines together by replacing the attribute mark between them with another character.

A solution for Wish #19, preventing terminal logons: This problem is partially solved on the new Microdata releases by incorporating SET-TIME and SET-DATE into the reboot. However, this doesn't help if there are other housekeeping chores that need to be completed. We have ten users on our system spread over a seven story building. I have addressed the problem by changing all the passwords in the SYSTEM file (except for SYSPROG) by the addition of a control character. I do this via a proc usable only from the SYSPROG account. At the same time, I modify the system LOGON message to show that we are "locked up" for maintenance. When done, I reverse this whole procedure with another proc. I have used this technique for three years of restores and other maintenance. I've done it while users were still logged on so that after they logged off, they couldn't log on again. I have never had any problems with it so far, but the manuals specifically warn programmers to be very careful when editing the SYSTEM pointers.

Al Rieland
WHA Computer Services
Madison, WI

Your solution to wake a sleeping process is similar to the serial printer driver mentioned in this issue's interview with Steve Kowarsky. It's unfortunate programmers have to go to so much trouble to make up for no WAKEUP verb. As for the Editor's prestore command capability, we also received a number of calls reminding us that a prestored command string could end with the P command itself, thereby causing an infinite loop that can repeatedly execute the string of commands. Actually, the Wish was asking for the capability of entering something like "3P", which would mean "execute the prestored command string only three times". We've tested your line splitting technique and you're right — it's easier

than it looks. For a hardware solution to preventing terminal logons, see Pragma #2, page 36. And finally, the warning on page 68 of Microdata's latest Programmer's Reference Manual would seem to imply that if the ITEM command shows that an account definition item in the SYSTEM file is not followed in its group by any logged on account definition items, then it's safe to edit the item.

—Editors

More on Computing Modulos

I have to take exception to your editorial reply to Joseph Zeligs' letter on the "Computing Modulos with ENGLISH" article (Pragma #1, page 29). Joseph Zeligs' letter is in Pragma #2, page 35.

You question what algorithm Mr. Zeligs prefers for computing modulos. The question is irrelevant. Mr. Zeligs did not publish an article on computing modulos with ENGLISH. You did. If you insist that a pre-requisite for criticizing any problem solution is to come up with an alternative solution to the problem, I fear that you will shortly discourage criticism of any problem solutions. Mr. Zeligs and myself sell the File Sizing Tool, a computer program which does resize files and uses a different algorithm than the one you described. The operation of the algorithm has been described in another Pick system journal.

You claim that the intent of the article was not to promote any file sizing algorithm as being preferable to any other. This is a self-serving declaration. The title of the article was, "Computing Modulos with ENGLISH". If, as you claim, the article was intended to, "show that ENGLISH is quite capable of performing complicated computations", you should have titled it as such. The title, "ENGLISH Capable of Complicated Computations", to paraphrase your claim, isn't exactly news to Pick users. To imply that the purpose of the article was anything other than to show people how to resize files is demagoguery.

In the second paragraph of the editorial reply, you state that determining the modulo is largely a matter of trading disk space for access time. This is true, but the space-time tradeoff is heavily weighted in favor of either space or time depending on the range of the modulus, the average item size, and the total number of items. Since you chose to dispute the amount of disk space used, I will discuss disk space utilization.

Mr. Zeligs states that a file with an average item size of 300 bytes should be sized in such a manner as to contain an average of 1 item per group. Your reply states that this will cause each group to contain 200 unused bytes resulting in wasted disc space. Your reply stands by the original algorithm, recommending that the file be sized to contain 1.66 items per group. In this manner, you hope to utilize disc space more efficiently. This is wrong.

Having 1.66 items in a group is like having 2.2 children. Although 1.66 is the average figure, it can't exist in real life. Groups hold only whole items. With files of average items sizes of 300 bytes, your groups will typically hold 1 or 2 items. If they hold 1 item, you're wasting the same 200 bytes that Mr. Zeligs is. If they hold 2 items, you are using 2 frames, one in overflow, and wasting 400 bytes of the second frame.

The reply goes on to criticize Mr. Zeligs' letter for "hypothesizing with nice round numbers or ignoring the effect of one or two factors". Mr. Zeligs used round numbers to easily illustrate a principle. The principle, that your algorithm wastes disc space, is correct whether you use nice round numbers or not. The other factors Mr. Zeligs chose to ignore, divisibility by two or five and the quality of the hashing distribution, likewise have no relevance to this discussion.

One last comment on your editorial policy. Give your authors (and your editors!) credit for writing articles. Some of us do travel to conventions and such and it would be nice to know who writes all this stuff.

Brian Gulino
Generation Research
San Diego, CA

We certainly don't "insist" on alternative solutions, otherwise we could not have published Mr. Zeligs' letter. But now we are more curious than ever, and we would still like to know what your algorithm is, or in which journal it can be found. After carefully re-reading the original article, we still believe it made no attempt to promote any algorithm as being preferable. The opening sentence quite clearly set the theme of using ENGLISH instead of BASIC, and nowhere was the algorithm itself explained or examined. Our response to the idea of one item per group explained that if that rule is followed even when the example file grows to 10,000 items, then two megabytes are wasted, but reducing the modulo from 10,000 to 3,333 would result in 3 items per group (not 1 or 2), thereby wasting only 100 bytes (not 400) of the second frame. Our "hypothesizing" remark applied to our own original reply as much as it did to Mr. Zeligs' letter; it was a bridge that led the reader from our own round numbers, with which we were hypothesizing and knew were unproven, to our final recommendation of using benchmarks — a recommendation that still stands. As for bylines, this is the first issue of Pragma containing articles from sources other than Semaphore Corporation, and those two pieces both identify their authors. Everything else we have ever published was created solely by our tiny writing staff, who have preferred to remain anonymous.

—Editors

ACCU/PLOT Upgrades

I would like to thank you for the help you have given in getting our advertisement into Pragma. Can we inform ACCU/PLOT users interested in receiving ACCU/PLOT-II to please contact AccuSoft Enterprises immediately? Also, AccuSoft Enterprises will be presenting seminars on the "Practical uses of ACCU/PLOT-II" during Pick Spectrum '83, to be held at Lake Tahoe on March 22 through March 26.

Mike Schellenbach
AccuSoft Enterprises
Los Angeles, CA

Generating Blank Forms

by Henry Wudarczyk
ECRM
Bedford, MA

Presented are examples of a simple technique for generating forms without writing code.

Formatting can make the difference between a good form and a poor one. By using string constants and adjusting the output parameters for dictionary words, some nice effects for forms can be created that might otherwise require some programming. See the examples below.

The number of boxes that can be formed by one dictionary word is limited by the number of characters that can be stored in the F-stack. Begin testing the limit with about seven boxes. The technique is very easy to experiment with: try altering the characters that form each box, or try mixing different spacings or sizes of boxes.

```
    BL
001 A
002 99
003
004
005
006
007
008 F;C¦        ¦¦_______¦
009 L
010 8
```

The correlative at left would normally be output exactly as defined in attribute eight. But by left justifying the string data and limiting the field width to half of the string size (a field width of 8 for a string of 16 characters), the string is too large for the field. It automatically wraps around within the column, resulting in a box formation as shown on the right.

```
BL. . . . . .
¦        ¦
¦_______ ¦
¦        ¦
¦_______ ¦
¦        ¦
¦_______ ¦
¦        ¦
¦_______ ¦
```

```
    BLOCK
001 S
002 99
003    A        B        C        D
004
005
006
007
008 F;C¦        ¦        ¦        ¦        ¦¦_______¦_______¦_______¦_______¦
009 L
010 29
```

```
   A        B        C        D

¦        ¦        ¦        ¦        ¦
¦_______ ¦_______ ¦_______ ¦_______ ¦
¦        ¦        ¦        ¦        ¦
¦_______ ¦_______ ¦_______ ¦_______ ¦
¦        ¦        ¦        ¦        ¦
¦_______ ¦_______ ¦_______ ¦_______ ¦
¦        ¦        ¦        ¦        ¦
¦_______ ¦_______ ¦_______ ¦_______ ¦
```

The one column technique can be modified to form a complete matrix of boxes, simply by repeating first the pattern for the open top of each box, then the pattern for the bottom of each box, as shown in this example.

games

For many people, computers are synonymous with games, now that the video game industry has become such a giant. Programmers frequently cut their teeth on small game programs, since such programs are often straight-forward and self-contained without a lot of complicated interfaces to files or other software. And, more than anything, games are simply entertaining and fun.

The Games Department will be making periodic appearances in Pragma. If you have a game program you would like to share, send it in for publication. If there's anything the Pragma staff has time to do, it's "performing rigorous software quality assurance" (in other words, trying out a new game on the computer).

Animal, a Program that Learns

Ever play "Animal, Vegetable, Mineral"? Think of an animal. How many questions might someone have to ask you to figure out what animal you're thinking of? Do obscure animals require more guesses? What guesses are necessary if the animal is mythical, or if you just invented the animal in your mind? Can a program be designed to somehow guess any animal you might imagine? Does the program have to only know about animals, or can it try to guess whatever it is you're thinking about?

Animal is a program that tries to guess what animal you're thinking of. It remembers new animals that it hasn't heard of, so that it can make better guesses the next time it tries.

Animal relies on a growing file of items. Each item consists of three attributes:

AMC 0, a question to ask.
AMC 1, the next question to ask if the answer to AMC 0 was "yes".
AMC 2, the next question to ask if the answer to AMC 0 was "no".

(AMC stands for Attribute Mark Count, or simply Attribute.) Sometimes, AMC 0 is used by the Animal program to make a final guess with no follow-up questions. In that case, AMC 1 is simply an asterisk (*), and AMC 2 is null (empty). For example, here are two items (the minimum required) making up a complete Animal file:

ARE YOU THINKING OF AN ANIMAL (YES OR NO)
IS IT A LION
ARE YOU THINKING OF AN ANIMAL (YES OR NO)

IS IT A LION
*

The *root* item is always used by Animal to start the questioning. In a file with many items, the items actually form a binary tree, with paths leading from the root to every question in the file, since any question appearing in AMC 1 or AMC 2 is also in AMC 0 of some item.

To play the game, the program starts with the root question, and then asks subsequent questions as determined by the player's yes or no answers. If an asterisk is reached as the program traverses the binary tree file of questions, the program has "guessed" the player's animal. But if a null is reached, the program has failed, and the player is asked for a new animal to be added to the file. Here is a sample dialogue between the program and a player. The player's input is underlined:

```
:RUN BP ANIMAL
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?NO
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?YES
DOES IT LIVE UNDER WATER?NO
DOES IT HAVE WINGS?NO
DOES IT HAVE LEGS?YES
DOES IT HAVE FOUR LEGS?YES
DOES IT EAT MEAT?YES
IS IT A LION?NO
I GIVE UP! WHAT IS YOUR ANIMAL?A DOG
BEFORE I ASK 'IS IT A LION?',
WHAT NEW QUESTION CAN I ASK THAT WILL UNIQUELY IDENTIFY
A DOG?IS IT A COMMON HOUSE PET
WHEN THINKING OF A DOG,
WHAT IS THE ANSWER TO 'IS IT A COMMON HOUSE PET'?YES
I'LL REMEMBER THAT.
DO YOU WANT TO PLAY AGAIN?YES
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?YES
```

NO MORE WAITING FOR P. & L.'S

IF YOU HAVE

ADDS Mentor

DEC Ultimate

Honeywell Ultimate

IBM Series 1

Microdata

Prime Information

or any PICK operating system,

this is the best* financial package you can buy. GAAP™ has a full warranty, complete documentation, password security control, and free upgrades for 1 year.

GENERAL LEDGER

With GAAP™, you can have financial reports every day of the year. Year-to-date, period-to-date balance and budget analysis are a simple selection from the software menu. Capabilities include:

Chart of Accounts Maintenance • Journal Vouchers Entry • Recurring Journal Vouchers Entry • Journal Voucher Reports • Cash Receipts Entry • Trial Balance(s) • Statements (P & L, Bal/Sh., etc.) • Transaction Registers • Chart of Accounts Analysis (Audit Trails) • Chart of Accounts Listings • Statement Format Table Listings • General Ledger Summary Display • Suspended Batch Report • Summary Income/Expense Reports

ACCOUNTS PAYABLE

Cash or accrual accounting, automatic check issuance, plus:

Voucher Entry and Inquiry • Voucher Registers • Cash Requirements Payment Schedules • Voucher Payment Selection Process • Issue Checks Process (Original issue) • Reissue Checks Process • Reset Check MICR Numbers • Manual Disbursements Posting • Void Checks Process • Cash Disbursements Journals • Check Registers • Check Reconciliation Process • Vendor Analysis (Audit Trail) • Vendor File Listings • Vendor File Maintenance • Bank Code File Maintenance • Terms File Maintenance

ACCOUNTS RECEIVABLE

Produces ready-to-mail statements on a balance forward or open invoice basis. Functions include:

A/R Transaction Posting • A/R Payment-on-Account Application • Customer Open Invoices Listing • Customer Analysis • Aged Trial Balance • Finance Charge Calculation • Statement Printing • Delinquent Aged Trial Balance • Delinquent Statement Printing • Open Invoice Listing • A/R Transaction Registers • Zero Balance Invoice Removal • Customer File Maintenance

OTHER OPTIONS

- Purchasing/Receiving Interface
- Inventory Control Interface
- Custom Software

FREE TRIAL

We believe in our product so much that we want you to try it for 30 days. In fact, we guarantee a full refund, no questions asked, if within 90 days you determine GAAP™ does not meet your needs.

Turn your accounting system into a management tool, every day of the year! Call us today for more information.

KDK Enterprises, Inc.

Suite 302
1491 Chain Bridge Road
McLean, VA 22101
703/893-7883

*A Washington, D.C. auditing firm found that GAAP™, as the name implies, truly follows Generally Accepted Accounting Principles.

GAAP™

PRAGMA • ISSUE 3 • FEBRUARY, 1983 • 207 Granada Drive • Aptos, CA 95003

TYPICAL ANIMAL DATA FILE

QUESTION.....	NEXT QUESTION IF YES.....	NEXT QUESTION IF NO.....
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)	DOES IT LIVE UNDER WATER	ARE YOU THINKING OF AN ANIMAL? (YES OR NO)
DOES IT EAT MEAT	IS IT A COMMON HOUSE PET	IS IT A COW
DOES IT FLY	IS IT A BIRD OF PREY	IS IT A CHICKEN
DOES IT HAVE A SHELL	DOES IT HAVE PINCERS	DOES IT HAVE TENTACLES
DOES IT HAVE A STINGER	IS IT A BEE	IS IT A FLY
DOES IT HAVE FEATHERS	DOES IT FLY	IS IT AN INSECT
DOES IT HAVE FINS	DOES IT HAVE GILLS	DOES IT HAVE A SHELL
DOES IT HAVE FOUR LEGS	DOES IT EAT MEAT	DOES IT HAVE TWO LEGS
DOES IT HAVE GILLS	DOES IT LIVE ONLY IN FRESH WATER	DOES IT LIVE ONLY IN SALT WATER
DOES IT HAVE LEGS	DOES IT HAVE FOUR LEGS	IS IT A TYPE OF SNAKE
DOES IT HAVE PINCERS	IS IT A CRAB	IS IT A SHRIMP
DOES IT HAVE TENTACLES	IS IT AN OCTOPUS	IS IT A STARFISH
DOES IT HAVE TWO LEGS	IS IT A MAN	IS IT A CENTIPEDE
DOES IT HAVE WINGS	DOES IT HAVE FEATHERS	DOES IT HAVE LEGS
DOES IT LIVE IN A SHELL	IS IT A SNAIL	IS IT A WORM
DOES IT LIVE ONLY IN FRESH WATER	IS IT A TROUT	IS ITS MEAT COMMONLY EATEN
DOES IT LIVE ONLY IN SALT WATER	IS IT A TYPE OF WHALE	IS IT A MANATEE
DOES IT LIVE UNDER WATER	DOES IT HAVE FINS	DOES IT HAVE WINGS
IS IT A BAT	*	
IS IT A BEE	*	
IS IT A BIRD OF PREY	IS IT AN EAGLE	IS IT A ROBIN
IS IT A CENTIPEDE	*	
IS IT A CHICKEN	*	
IS IT A COMMON HOUSE PET	IS IT A DOG	IS IT A LION
IS IT A COW	*	
IS IT A CRAB	*	
IS IT A DOG	*	
IS IT A FLY	*	
IS IT A KILLER WHALE	*	
IS IT A LION	*	
IS IT A MAN	*	
IS IT A MANATEE	*	
IS IT A PYTHON	*	
IS IT A RATTLESNAKE	*	
IS IT A ROBIN	*	
IS IT A SALMON	*	
IS IT A SEA TURTLE	*	
IS IT A SHARK	*	
IS IT A SHRIMP	*	
IS IT A SNAIL	*	
IS IT A STARFISH	*	
IS IT A TROUT	*	
IS IT A TYPE OF SNAKE	IS IT POISONOUS	DOES IT LIVE IN A SHELL
IS IT A TYPE OF WHALE	IS IT A KILLER WHALE	IS IT A SEA TURTLE
IS IT A WORM	*	
IS IT AN EAGLE	*	
IS IT AN INSECT	DOES IT HAVE A STINGER	IS IT A BAT
IS IT AN OCTOPUS	*	
IS IT POISONOUS	IS IT A RATTLESNAKE	IS IT A PYTHON
IS ITS MEAT COMMONLY EATEN	IS IT A SALMON	IS IT A SHARK

```
DOES IT LIVE UNDER WATER?NO
DOES IT HAVE WINGS?NO
DOES IT HAVE LEGS?YES
DOES IT HAVE FOUR LEGS?YES
DOES IT EAT MEAT?YES
IS IT A COMMON HOUSE PET?YES
IS IT A DOG?YES
I GUESSED YOUR ANIMAL!
DO YOU WANT TO PLAY AGAIN?NO
```

A typical data file resulting from the above input is shown in the listing on page 42. The listing on page 46 is the complete source code for the Animal game as written in DATA/BASIC for the Microdata.

Line 1 opens the file named "ANIMALS" for subsequent access by READ and WRITE statements. All data items exist in this one file.

Lines 3 to 47 form one large LOOP which is executed once for each time the game is played. The LOOP REPEATs until the variable PLAY is set to "NO" in line 46. PLAY is initialized to "YES" in line 2 to guarantee the LOOP is executed at least once.

Lines 7 to 15 form a LOOP that the computer REPEATedly executes as it travels down the tree of questions. The variable ROOT contains the current question being asked; it is initialized to a known starting value in line 6. The program assumes the file has at some time been initialized to contain at least two items, like the two minimum required items previously mentioned. One item has an identifier set to the standard question used for initializing the ROOT variable, and the other item contains a final guess. Since INPUT statements automatically prompt with a "?", question marks are not actually stored at the end of each question in items. Once the file has been initialized with two such items, the game can be played any number of times, causing new items to be added to the file.

The PARENT (initialized in line 5) and GRANDPARENT variables are used to maintain a history of the questions asked, in case the program later needs to insert a new animal and question into the appropriate place in the tree file. The aging of PARENT to GRANDPARENT and ROOT to PARENT occurs in lines 12 and 13.

Line 10 sets the ANSWER variable (initialized in line 4) to the player's "YES" or "NO" response, which is solicited after the program's current guess is made in line 9. OLDANSWER remembers a previous ANSWER for possible later use in line 41, when the program needs to know whether to replace a left or right tree branch.

AMC 0 of every item is the identifier by which the remaining data in AMC 1 and AMC 2 can be accessed. Line 11 READs an item from the file. In line 14, the program sets ROOT to the next question to ask, which is either AMC 1 or AMC 2 of the item just read. The attribute selected depends on the player's ANSWER. The LOOP is then REPEATed with the new ROOT.

When the program finally extracts as asterisk or a null, the WHILE condition in line 7 will no longer hold, the LOOP terminates, and line 16 is reached. If the final ROOT is an asterisk, the program has won and reports success, ELSE lines 17 to 44 are executed in order to learn a new animal.

The player types in the name of the new animal when line 18 is executed. To simplify the code, the program assumes the player will provide the necessary indefinite article and avoid any terminating punctuation. In other words, the program expects input like "A CHICKEN" or "AN OSTRICH" with no ending period.

Line 19 creates a new file item for the new animal. The asterisk forms AMC 1, AMC 2 is null, and the item's identifier in AMC 0 is formed by prefixing "IS IT" to the animal's name. (What happens if the new animal is actually already somewhere else in the file?)

Whenever a new animal is added to the file, a new question must also be added that will allow the program to differentiate between the new animal and the last guess it made. Lines 21 to 23 ask for this, but in an awkward way. A better prompt might be something like WHAT QUESTION CAN I ASK TO TELL A COW FROM A HORSE?, but this would require temporarily stripping the "IS IT" from the PARENT. The current phrasing is used to keep the code uncluttered and hopefully more understandable. For the same reasons, the program assumes the player will type in a sufficiently short question (remember the 50 character limit on item identifiers) with no terminating question mark or other special punctuation in response to line 24.

The LOOP in lines 20 to 28 accepts questions from the player until one is found which is not already used in the data file for distinguishing between two already-known animals. A totally new question is needed since the program will make one of its tree branches (AMC questions) point to the new animal, and existing AMC 0 questions that aren't a final guess already have both their branches (AMC 1 and 2) used up.

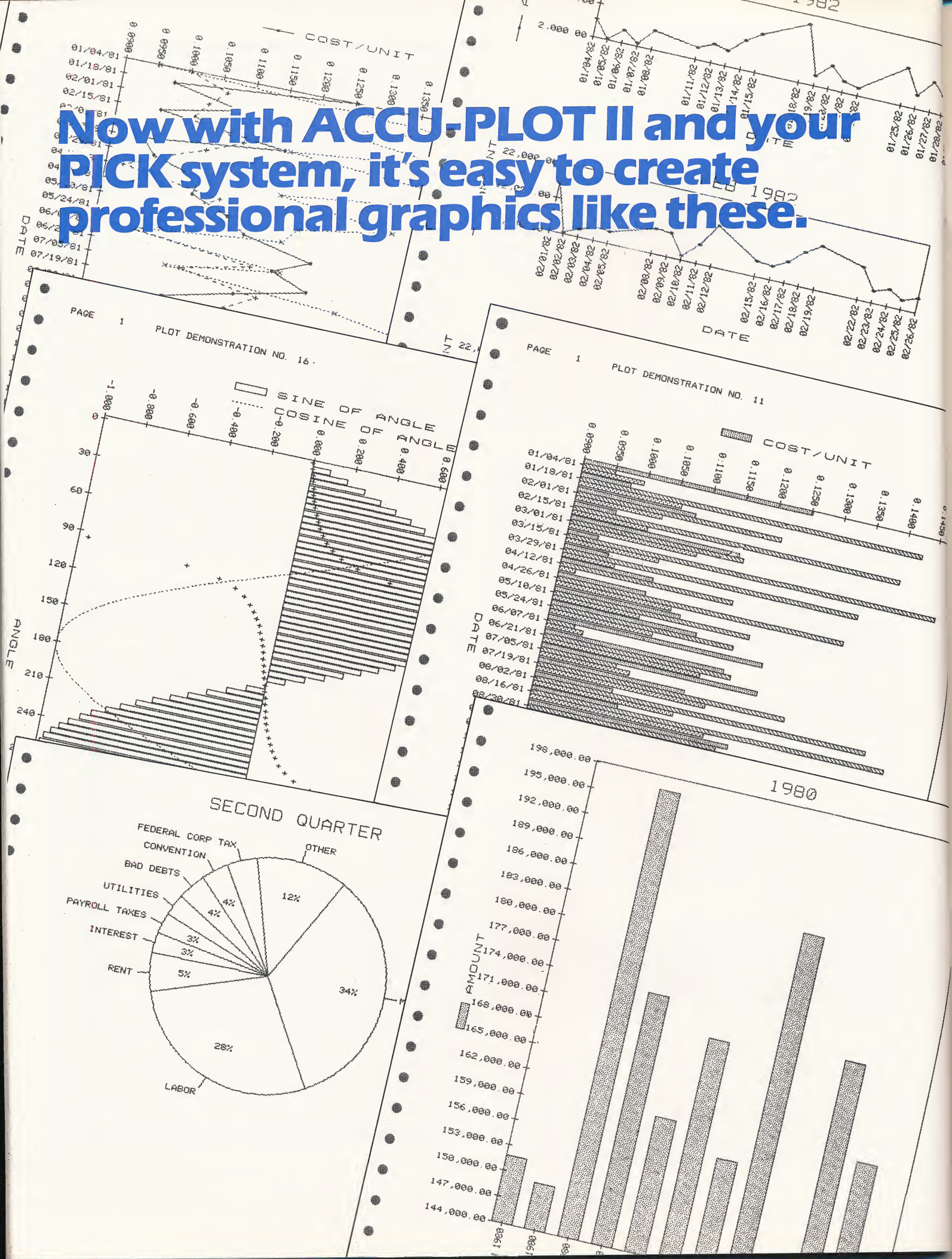
Line 25 attempts a READ of each proposed new question. If the READ fails, then the question has not been used, the ELSE clause is executed to null ITEM, the WHILE test in line 26 will fail, and line 29 will be reached.

Lines 29 to 31 determine which AMC of the new question item should point to the new animal. The actual question item is built by lines 33 and 34 or 36 and 37, and created by line 39.

Lines 40 to 42 complete the insertion of the new question into the tree file. The READ in line 40 fetches the item pointing to (containing the identifier of) the program's last guess. The pointer is then replaced with a pointer to the new question. The patched GRANDPARENT item is then written back out to the file to complete the insertion of the new question. In this way, the program allows the file to continually grow for future players, with no other special intervention on the part of the user.

Animal is a fascinating game to play on a computer, since the machine seems so intelligent and participates in such a lively way. Of course, since the program is written in a completely general fashion, there's no reason why the data file has to be limited to animals. The program can just as easily be called "Mind Reader", the initial ROOT question can simply be ARE YOU THINKING OF SOMETHING? (with the second minimum item set to any type of final guess, like IS IT CHOCOLATE ICE CREAM?), and the program logic will still function perfectly as it tries to guess whatever the player is thinking. Another possibility is to change Animal into a general learning program to help diagnose machine failures. In that case, the program can be set up to use a data file similar to the Trouble Tree presented in Pragma #2 on page 32. As new machine failures are encountered, the program could provide a convenient mechanism for inserting the failures' descriptions and repair solutions into the Trouble Tree. P

Now with ACCU-PLOT II and your PICK system, it's easy to create professional graphics like these.



When it comes to communicating complex business information, an ACCU-PLOT picture is worth 1,000 words.

ACCU-PLOT II from Accu/Soft Enterprises is a new, easy to use graphics system that will produce bar charts, point-to-point line charts, scatter graph diagrams and pie charts in either black and white or color.

ACCU-PLOT II is as easy to use as ENGLISH. The sentences used to produce graphs are similar to the "LIST" and "SORT" verbs you use every day. No modification to your dictionaries or data files is required.

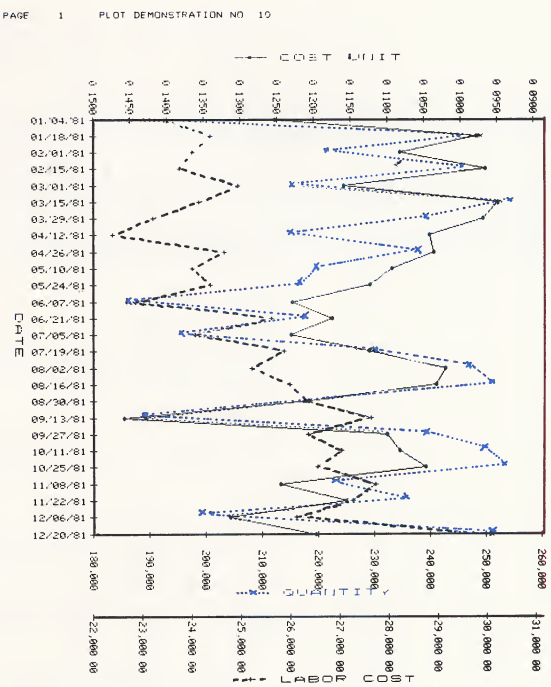
ACCU-PLOT II is available for Microdata, Ultimate, ADDS Mentor, Evolution, CDI's IBM Series/I, Datamedia, General Automation and other PICK based computer systems. ACCU-PLOT II will operate on most dot matrix printers with graphics capabilities as well as most graphic CRT's, including the IDS Prism printer and other devices capable of producing color graphics.

Some of the key features of ACCU-PLOT II are:

- Syntax similar to the standard inquiry language formats (ENGLISH, RECALL, INFO/ACCESS, etc.)
- Standard data files and dictionaries.
- Any number of attributes can be plotted.
- Line, bar, or scatter formats may be mixed.
- Pie charts can show "exploded" slices.
- Automatic scaling with optional override.
- User-specified captions above and below graph.
- Horizontal and vertical formats are available.
- Multiple graphs can be placed on a single page.
- A single graph may be spread over multiple pages.
- Text data may be placed on the x-axis.
- Y-axis attributes may be grouped for common scaling.
- User definable line and bar styles and chart format.
- Four character sizes for dot matrix printers.
- Interfaces with BASIC programs for special applications.
- Interfaces with COMPU-SHEET electronic spreadsheet.

ACCU/SOFT ENTERPRISES
8655 Belford Avenue, Suite 100
Los Angeles, California 90045

Here's how easy it is to create this chart:



Simply enter: SPLOT PLOTDEMO2 BY DATE ID-SUPP DATE COST/UNIT QUANTITY LABOR-COST LPTR HEADING "PAGE" 'P' PLOT DEMONSTRATION NO. 10 'LL' "

Put the graphic power of ACCU-PLOT at your command. For more information, mail this coupon or call (213) 649-5800 today.

Yes! Please send me more information about ACCU-PLOT II.

Name _____
Title _____
Company _____
Address _____
City _____
State _____ Zip _____
Telephone ( ) _____

ANIMAL PROGRAM LISTING

```
001 OPEN "ANIMALS" ELSE PRINT "ANIMALS FILE IS MISSING!" ; STOP
002 PLAY = "YES"
003 LOOP UNTIL PLAY = "NO" DO
004    ANSWER = ""
005    PARENT = ""
006    ROOT = "ARE YOU THINKING OF AN ANIMAL? (YES OR NO)"
007    LOOP WHILE (ROOT # "*") AND (ROOT # "") DO
008       OLDANSWER = ANSWER
009       PRINT ROOT:
010       INPUT ANSWER
011       READ ITEM FROM ROOT ELSE PRINT "ANIMALS FILE IS DAMAGED!" ; STOP
012       GRANDPARENT = PARENT
013       PARENT = ROOT
014       IF ANSWER = "YES" THEN ROOT = ITEM<1> ELSE ROOT = ITEM<2>
015    REPEAT
016    IF ROOT = "*" THEN PRINT "I GUESSED YOUR ANIMAL!" ELSE
017       PRINT "I GIVE UP!  WHAT IS YOUR ANIMAL":
018       INPUT NEWANIMAL
019       WRITE "*" ON "IS IT ":NEWANIMAL
020       LOOP
021          PRINT "BEFORE I ASK '":PARENT:"?',"
022          PRINT "WHAT NEW QUESTION CAN I ASK THAT WILL UNIQUELY IDENTIFY"
023          PRINT NEWANIMAL:
024          INPUT QUESTION
025          READ ITEM FROM QUESTION ELSE ITEM = ""
026       WHILE ITEM # "" DO
027          PRINT "SORRY...I ALREADY USE THAT QUESTION."
028       REPEAT
029       PRINT "WHEN THINKING OF ":NEWANIMAL:","
030       PRINT "WHAT IS THE ANSWER TO '":QUESTION:"'":
031       INPUT ANSWER
032       IF ANSWER = "YES" THEN
033          ITEM<1> = "IS IT ":NEWANIMAL
034          ITEM<2> = PARENT
035          END ELSE
036              ITEM<1> = PARENT
037              ITEM<2> = "IS IT ":NEWANIMAL
038              END
039       WRITE ITEM ON QUESTION
040       READ ITEM FROM GRANDPARENT ELSE PRINT "I'VE LOST MY MEMORY!" ; STOP
041       IF OLDANSWER = "YES" THEN ITEM<1> = QUESTION ELSE ITEM<2> = QUESTION
042       WRITE ITEM ON GRANDPARENT
043       PRINT "I'LL REMEMBER THAT."
044       END
045    PRINT "DO YOU WANT TO PLAY AGAIN":
046    INPUT PLAY
047 REPEAT
048 END
```

PRAGMA
207 GRANADA DRIVE
APTOS, CA 95003

BULK RATE
U.S. POSTAGE
PAID
APTOS, CA 95003
Permit No. 67

·ADDRESS CORRECTION REQUESTED